



Multi-Agent Learning for Security and Sustainability

Thesis submitted in accordance with the requirements of the University of Liverpool for
the degree of Doctor in Philosophy by

Richard Klima

October 2019

Abstract

This thesis studies the application of multi-agent learning in complex domains where safety and sustainability are crucial. We target some of the main obstacles in the deployment of multi-agent learning techniques in such domains. These obstacles consist of modelling complex environments with multi-agent interaction, designing robust learning processes and modelling adversarial agents. The main goal of using modern multi-agent learning methods is to improve the effectiveness of behaviour in such domains, and hence increase sustainability and security. This thesis investigates three complex real-world domains: space debris removal, critical domains with risky states and spatial security domains such as illegal rhino poaching.

We first tackle the challenge of modelling a complex multi-agent environment. The focus is on the space debris removal problem, which poses a major threat to the sustainability of earth orbit. We develop a high-fidelity space debris simulator that allows us to simulate the future evolution of the space debris environment. Using the data from the simulator we propose a surrogate model, which enables fast evaluation of different strategies chosen by the space actors. We then analyse the dynamics of strategic decision making among multiple space actors, comparing different models of agent interaction: static vs. dynamic and centralised vs. decentralised. The outcome of our work can help future decision makers to design debris removal strategies, and consequently mitigate the threat of space debris.

Next, we study how we can design a robust learning process in critical domains with risky states, where destabilisation of local components can lead to severe impact on the whole network. We propose a novel robust operator κ which can be combined with reinforcement learning methods, leading to learning safe policies, mitigating the threat of external attack, or failure in the system.

Finally, we investigate the challenge of learning an effective behaviour while facing adversarial attackers in spatial security domains such as illegal rhino poaching. We assume that such attackers can be occasionally observed. Our approach consists of combining Bayesian inference with temporal difference learning, in order to build a model of the attacker behaviour. Our method can effectively use the partial observability of the attacker's location and approximate the performance of a full observability case.

This thesis therefore presents novel methods and tackles several important obstacles in deploying multi-agent learning algorithms in the real-world, which further narrows the reality gap between theoretical models and real-world applications.

Acknowledgements

This thesis is the outcome of four very fruitful years, during which I have grown both professionally and personally. As any PhD student can probably confirm, one often faces ups and downs on this journey. The downs were bearable due to many great people who surrounded me during this episode of my life, the ups were shared and enjoyed also thanks to those people. My professional contacts have often transitioned into friendships, leading to a great work environment, which I really enjoyed to be part of.

Firstly, I am very grateful to you, Karl, you have been a great supervisor, incredible inspiration and an amazing person to work with, steering me through this unforgettable life experience. I have learned a lot from you and your valuable feedback has been great. I would like to also express my gratitude to you, Rahul, my second supervisor, you were a great support when needed and extremely pleasant to work with. And you, Frans, thank you for introducing me to the world of POMDPs. I am also very thankful to you, Daan, I learned a lot from you and had great collaboration with you throughout my PhD, I could always bounce my research ideas off you. I am also grateful to the researchers from the European Space Agency, Daniel, Dario, Alex and others who I worked with on the space debris removal project. You provided great insights into the space debris removal problem and helped me greatly to develop the simulator. I also need to mention my wonderful visit at CWI in Amsterdam, working with you, Daan (again) and Michael. You were both extremely easy and pleasant to work with. I learned a lot from you not only professionally.

During my studies I was fortunate enough to attend several conferences to present my work and to meet some incredible people, many of whom I had known from reading academic papers first, for which I am very grateful to the department for making it possible.

It was very important for me that I had plenty of opportunities to take my mind from work time to time, be it sometimes going for a pint on Friday after work or trips around the UK and Europe. To name some of you, Paul, Maryam, Elisa, Francesco, Sven, Fabio, Rafael, Angelo and other colleagues from the department, the archaeologists and all the others, thank you for all the fun we have had.

Above all my greatest gratitude needs to go to my family, foremostly to you, my parents Jiřina and Petr, you have always supported me no matter what and keep believing in me.

This thesis is primarily my own work. The sources of other materials are identified. However, we all *stand on the shoulders of giants*.

Contents

Abstract	i
Acknowledgements	iii
Contents	v
1 Introduction	1
1.1 Multi-Agent Learning for Security and Sustainability	2
1.2 Problem Statement and Research Questions	5
1.3 Contributions and Thesis Outline	11
2 Preliminaries	13
2.1 Game Theory	15
2.1.1 Normal-Form Games	15
2.1.2 Solution Concepts	16
2.1.3 Evolutionary Game Theory	18
2.1.4 Multi-Stage Games	19
2.2 Reinforcement Learning	22
2.2.1 Markov Decision Process	24
2.2.2 Policy & Value Function	25
2.2.3 Temporal Difference Learning	26
2.2.4 Partially Observable Markov Decision Process	28
2.3 Multi-Agent Learning	29
2.3.1 Stochastic Games	30
2.3.2 Approaches to Multi-Agent Learning	31
3 Methodology & Application Areas	35
3.1 Security and Sustainability in Studied Domains	36
3.1.1 Sustainability of Earth's Orbit	36
3.1.2 Safety in Critical Systems with Risky States	37
3.1.3 Mitigating Threats in Spatial Security Domains	38
3.2 Methodology and Problems Classification	39
3.2.1 Threat Types	40

3.2.2	Modelling Choices	41
3.2.3	Input Data	43
4	Modelling and Learning in the Space Debris Removal Problem	45
4.1	Space Debris Removal Problem	46
4.2	Simulating Space Debris Environment	50
4.3	Models of Agent Interaction	51
4.3.1	Surrogate Model of the Space Debris Simulator	52
4.3.2	Deterministic Game Model of Space Debris Removal	54
4.3.3	Dynamic Decision Making in the Space Debris Problem	58
4.4	Evaluation of Different Models of Agent Interaction	61
4.5	Discussion	76
5	Robust Learning in Critical Systems with Risky States	80
5.1	Need for Robustness in Systems with Risky States	81
5.2	The Robust Temporal Difference Operator κ	84
5.2.1	Formal Model	84
5.2.2	Advanced Model	86
5.2.3	Examples of TD(κ) Methods	86
5.3	Theoretical Analysis of Convergence of Operator κ	89
5.3.1	Convergence to the Optimal Q^*	89
5.3.2	Convergence to the Robust Q_κ^*	93
5.3.3	Convergence in the Multi-Agent Case	95
5.4	Experiments with Robust Learning	96
5.4.1	Performance	97
5.4.2	Different Levels of Attack	99
5.4.3	Robustness Analysis	101
5.5	Discussion	102
6	Learning Against Adversarial Agents in Spatial Security Domains	104
6.1	Uncertainty in Spatial Security Domains	105
6.2	Partially Observable Model of Spatial Security Games	109
6.2.1	Observability in Spatial Security Games	110
6.2.2	Statistical Approach to Uncertainty	112
6.3	Q-learning with Bayesian Inference: BayesRQ	116
6.4	Experiments with the BayesRQ Algorithm	118
6.5	Discussion	122
7	Conclusion	124
7.1	Contributions and Answers to the Research Questions	124
7.2	Limitations and Perspectives for Future Research	128

List of Figures	132
List of Tables	137
References	141
Publications	157
Appendix A Space Debris Simulator Model	158
A.1 Simulating Space Debris Environment	158
A.2 Collision and Breakup Model	159
A.3 Simple Launch Model	160
A.3.1 Repeating Launch Sequence	160
A.3.2 Validation	163
A.4 Complex Launch Model	164
A.4.1 Mass per Year	165
A.4.2 Spacecraft Classes	166
A.4.3 Orbits	168
A.4.4 Launches	170
A.4.5 Future Launch Scenarios	170
A.4.6 Mega Constellations	171
Appendix B The Space Debris Removal Problem: Preliminary Work	176
B.1 Defining the Space Debris Removal Game	176
B.2 Simulation Results and Projections	178
B.2.1 Debris Evolution	179
B.2.2 Risk Evolution	179
B.3 Game Theoretic Analysis of Equilibria	181
B.3.1 Two Player Game	181
B.3.2 Strategic Substitutes and Existence of Pure Equilibrium	183
B.3.3 Evolutionary Dynamics	186
B.3.4 Three Player Game	188
B.4 Discussion	190
Appendix C Surrogate Model of Space Debris Evolution	192
C.1 Validation of Surrogate Model	192

1

Introduction

In recent years we have witnessed an astonishing speed of technological progress, which has improved many aspects of our lives. While we get to enjoy many perks brought by technological advances, we must not forget about new challenges which come hand in hand. At the forefront of the progress has been artificial intelligence (AI), a fruitful domain of computer science, which aims to mimic human intelligence. The ultimate goal of AI is to tackle today's problems and introduce higher effectiveness into various aspects of human activities [Russell and Norvig, 2016]. An optimistic way to think about the future prospects of AI is as an extension of humans, not a substitution, where AI coexists with us and not threatens us, significantly enhancing our abilities. Before deploying AI techniques into real world and making them part of our everyday lives one needs to make sure they are used in a safe way and thus do not harm us or the environment. One of the frequently discussed problems of modern technology is its negative impact on the environment, be it Nature or the society. This issue is often described in terms of *security* and a closely related term *sustainability*, which are some of the most important attributes to consider, when making use of modern AI methods for real-world applications. This thesis studies how to deploy modern intelligent techniques to obtain effective behaviour in order to

improve security and sustainability, presenting new ways to approach and mitigate threats in complex environments. We investigate how to overcome modelling challenges when deploying AI state-of-the-art techniques.

The main assumption of this thesis is that many real-world environments can be described in terms of a multi-agent framework, where several entities interact with each other. Furthermore these entities, which we will call agents, make decisions which might depend on the other agents and a state of the environment. In this thesis we are particularly interested in such a multi-agent interaction and its complex dynamics. Moreover, we study how effective and safe behaviour can be attained in multi-agent systems, increasing the applicability of modern AI methods to real-world domains. In general, such behaviour can be obtained either by exact mathematical computation or by adaption by *learning from interaction*, which we extensively analyse and compare.

This thesis focuses on several domains, where we investigate the applicability of modern AI techniques with the emphasis on security and sustainability. We first study the domain of the space debris removal problem where future sustainability is a prerequisite for further technological progress, depending heavily on for example satellite communication or navigation. Next, this thesis investigates how a robust behaviour can be attained in critical domains with risky states, leading to increased security of such systems. Lastly, we focus on spatial security domains where potentially intelligent adversaries aim to attack some critical targets. We discuss the example of illegal rhino poaching problem, where there is a risk of species extinction, therefore introducing effective behaviour might improve sustainability of the species.

In this chapter we briefly introduce the main AI frameworks which we use to approach the studied security and sustainability domains; game theory, reinforcement learning and multi-agent learning. We then present the problem statement from which we derive 3 main research questions which we aim to answer in this thesis.

1.1 Multi-Agent Learning for Security and Sustainability

Sustainability is the ability to be maintained at a certain level or rate, often the term relates to avoidance of the depletion of natural resources. The relation between sustainability and security is very close; sustainability can be defined in terms of security as the process of securing environment against future degradation. On the other hand security can be seen as the process of sustaining environment against potential threats. In this thesis we

investigate modelling challenges of the multi-agent learning paradigm in several real-world application domains from the perspective of these two notions. The general goal can be defined as effectively reacting to threats with the aim of finding an effective behaviour in those domains.

Arguably, many real-world domains can be modelled as a multi-agent scenario, where the effect of every decision made by an agent also depends on other people (agents), which is especially true when we are concerned about safety and sustainability. The dynamics of multiple agents' behaviour can be very complex because they can influence each other, often in myriad unobservable ways. Thus, formalising the complex interactions of multiple agents in mathematical terms is a very difficult task. The fields of game theory, reinforcement learning and multi-agent learning (MAL) are trying to tackle this challenge. Tuyls and Stone [2018] describe multi-agent learning as a process where multiple agents learn to behave so that they achieve their goals, while they interact with other (learning) agents, who might have similar or different goals (e.g., cooperative or adversarial agents).

Depending on the application domain we might require different properties from the solutions, for example safety, robustness, convergence guarantees or sampling efficiency. We discuss several of such properties and their significance in order to achieve security and sustainability in the studied domains. This thesis combines and compares several modelling perspectives, which might be beneficial to overcome the difficulties when deploying multi-agent learning techniques [Albrecht and Stone, 2018]. Our multi-agent learning approach builds on using ideas from game theory and reinforcement learning, which provide powerful frameworks to model complex potentially multi-agent interactions. Moreover, the multi-agent learning paradigm has been developed in recent years primarily as an intersection of **game theory** and **reinforcement learning** [Tuyls and Weiss, 2012]. We now give a brief introduction to these fields.

Game theory models an interaction between rational agents (players), who aim to maximise their expected utility (reward). The field of game theory was extensively developed in 1940s and 1950s with the seminal works of Von Neumann and Morgenstern [1944] and Nash [1951], who set the stepping stones of modern game theory. The need for analysing agents' interactions comes from many real-world domains, where it has been successfully applied, examples include economics, politics, social science, biology, logic or computer science. A game-theoretical model often assumes several strong properties which do not always naturally appear in real-world agent interactions; such as pure rationality,

selfishness, or the concept of achieving behavioural equilibria among agents. Some of the limitations have been addressed to some extent by related fields of evolutionary game theory or multi-agent learning. This thesis draws ideas from these fields, proposing new methods to tackle challenges posed by real-world applications. Game theory offers a static solution concept in form of equilibria of strategies, which can be computed exactly by mathematical methods in simple cases. However in larger or dynamic systems such an exact computation might be unfeasible and we need to turn to adaptation by learning from interaction. This alternative solution approach is studied in the field of reinforcement learning. In this thesis we analyse and compare these two different approaches to obtaining effective behaviour.

Reinforcement learning (RL) is inspired by the way how humans learn from an early age, using trial and error [Lake et al., 2017]. For example if a child touches a hot object and gets hurt, receiving negative feedback (reward), it eventually learns not to touch it again, however if a child tastes ice-cream, receiving positive feedback (reward), it wants to do it again, because it brings pleasure. Similarly, in reinforcement learning we sample different experiences and receive corresponding rewards, which we use to update our knowledge about the environment, forming a value function. By interacting with the environment we learn what is good and what is bad for us in order to improve our future behaviour. Reinforcement learning is thus suitable when we seek optimal behaviour; for a general introduction to RL see the book of Sutton and Barto [1998]. Optimal behaviour can be defined in various ways depending on different metrics, in this thesis we are interested in behaviour which is safe to the environment while being effective and robust. Such properties are especially desirable for domains where we want to achieve sustainability. Reinforcement learning has seen many great success stories in recent years, to name a few: DQN [Mnih et al., 2013, 2015] combining RL with deep neural networks and achieving superhuman performance in Atari games, AlphaGo and AlphaZero [Silver et al., 2016, 2017] beating the best players of the game Go or successful application of RL to robotics [Abbeel et al., 2007; Kober et al., 2012; Levine et al., 2016].

The classic reinforcement learning is designed for single-agent domains. However, in many real-world applications we deal with multiple learning agents, thus single-agent reinforcement learning needs to be extended to multi-agent reinforcement learning (MARL), which introduces new complexities to the learning process. The main problem is non-stationarity caused by agents learning the behaviour of other learning agents; this is known as the *moving target* problem [Tuyls and Weiss, 2012]. This non-stationarity breaks

most convergence proofs we know from single-agent RL and makes obtaining theoretical guarantees in MARL extremely difficult.

In this thesis we use ideas from both, single-agent RL and multi-agent RL. The domains we focus on are naturally multi-agent systems but can be often abstracted and modelled also as a single-agent scenario. Arguably, when applying MARL methods it might be beneficial to start with an abstracted problem, investigating the use of single-agent RL methods to get better insights into the dynamics of the environment. In this thesis the main goal in the studied domains is to find an effective behaviour, often such behaviour cannot be computed exactly due to high complexity of environment, which is the reason we turn to learning the behaviour from interactions instead. Therefore in this work, where possible, we consider and compare the single- and multi-agent approach. Moreover, we investigate an exact computation of policies and learning of policies.

We can now introduce and define the problem statement and research questions which this thesis aims to answer.

1.2 Problem Statement and Research Questions

In this thesis we study how multi-agent learning and modelling techniques can be developed and applied to security and sustainability domains that require effective solutions, which are difficult to engineer by hand. We focus on three complex domains; the space debris removal problem, critical domains with risky states (e.g., smart power grids) and spatial security domains with adversarial agents (e.g., the illegal rhino poaching problem). Rather than taking a common bottom-up approach and starting from the algorithmic side, we take a top-down approach and start from the application domains.

Although the problem of learning in multi-agent settings has received significant attention in recent years, see for example surveys of Panait and Luke [2005]; Busoniu et al. [2008]; Bloembergen et al. [2015]; Hernandez-Leal et al. [2017]; Tuyls and Stone [2018]; Albrecht and Stone [2018], applying multi-agent learning techniques in complex domains remains largely an open problem with many challenges. Additionally multi-agent learning (MAL) inherits the problems we know from single-agent reinforcement learning such as the problem of *delayed reward* and *credit assignment*. In the multi-agent setting this is further compounded by complexities brought by non-stationarity caused by multi-agent interactions, known as the *moving target problem* [Tuyls and Weiss, 2012], which describes the situation where agents need to learn dynamically changing behaviour of other agents influencing the environment

states. Therefore, when applied in complex systems, effective multi-agent learning is an extremely difficult task. We call these issues the fundamental problems of MAL, which have been studied in isolation to some extent [Hernandez-Leal et al., 2017]. However, much less emphasis has been given to combinations of these fundamental problems, typically encountered when applying the methods into real world systems. The more common approach to multi-agent learning research is to start from very simple models and build up to more complex models aiming at tackling the fundamental problems, we choose a different perspective in this thesis. We start from complex real-world domains, asking the high-level question: how can we obtain optimal behaviour in such domains and what are the main obstacles in doing so. Therefore, this thesis does not aim to tackle the fundamental issues in isolation, but rather studies several complex problems, containing a combination of these issues, with the aim to develop specifically targeted solution methods. We focus on areas where security and sustainability is crucial, where we aim to additionally derive general principles that work in other domains as well.

As such this thesis aims to address the modelling obstacles of the multi-agent learning paradigm using game theory and reinforcement learning when applied to real-world complex domains. We describe such modelling obstacles in terms of 4 components of multi-agent learning as defined by Tuyls and Stone [2018]: (i) **the environment**, (ii) **the interaction mechanism**, (iii) **the learning mechanism** and (iv) **the agents**. We investigate three application domains where each domain predominantly contains one (two in the space debris removal problem) especially challenging MAL component (i-iv) to design. While highlighting the main challenging MAL component in each of the application domains, still all these domains require modelling of the other MAL components as well. Effectively addressing these modelling challenges will narrow the gap between theoretical models of MAL and their use in real-world applications. Overview of the problem statement is shown in Figure 1.1, where we state the modelling challenges of MAL, leading to a general approach to modelling complex environments. We now further describe the modelling challenges in turn together with the application domains.

When applying learning techniques to complex multi-agent domains we need to design a good model of such a multi-agent setting, which needs to be realistic enough while computationally viable.

“All models are wrong, but some are useful.”

—George Box

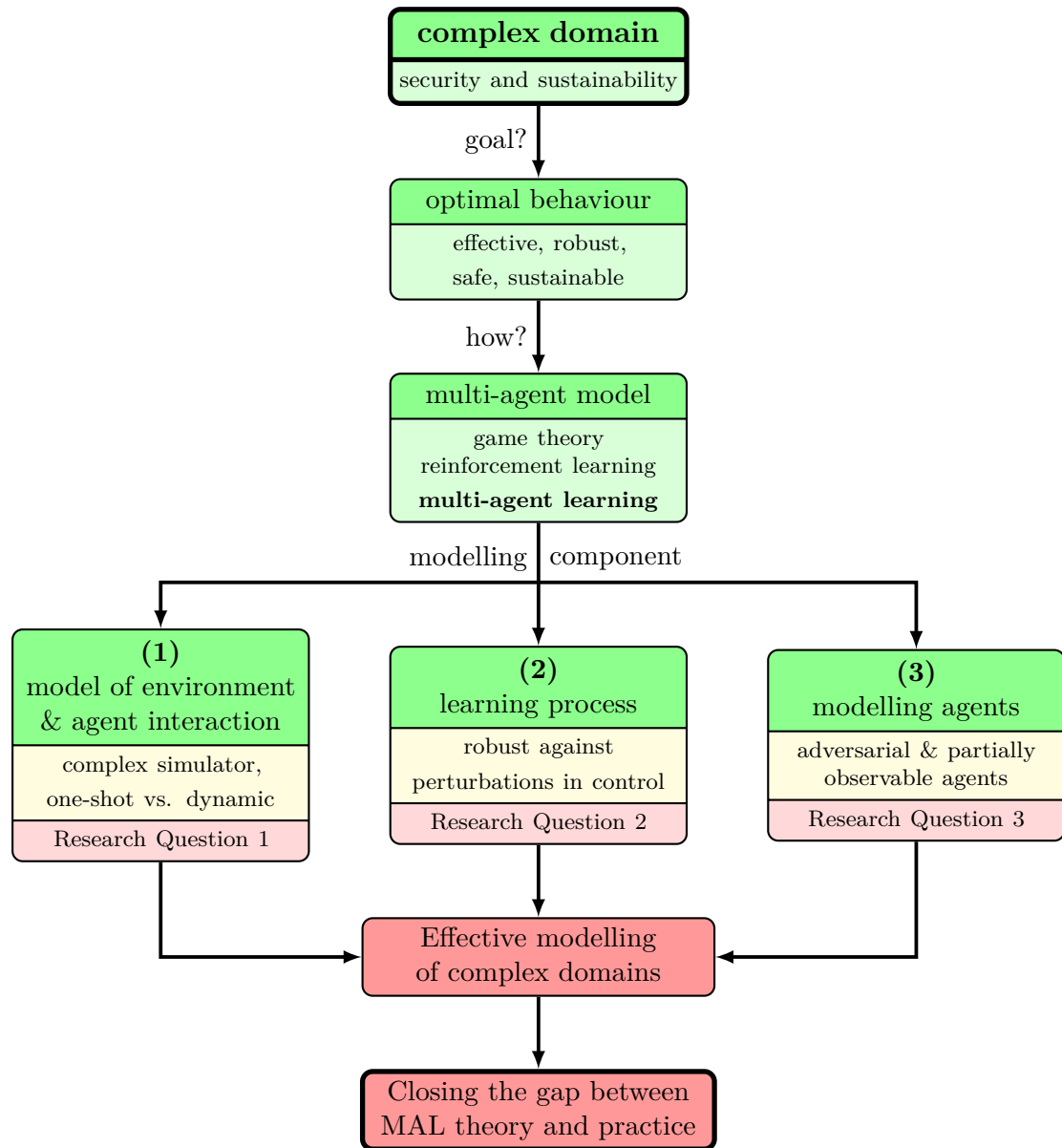


Figure 1.1: Problem statement structure.

(1) model of environment and agent interaction We study the challenge of modelling a complex environment in the domain of the **space debris removal problem**, where we propose several ways to model the environment. We investigate different modelling assumptions such as a one-shot interaction model compared to a dynamic multi-stage interaction model. Moreover, in many real-world domains the level of cooperativeness of agents might vary depending on many factors. It is thus important to analyse different models of agent interaction mechanisms, from cooperative to non-cooperative. We aim to propose a methodology comparing these different modelling approaches, evaluate their effectiveness and describe how realistic they are in the application domain. We also discuss how addressing these modelling issues can specifically contribute to tackling the space debris removal problem.

(2) designing learning processes Many real-world complex domains require a solution with certain features (e.g., robustness, fairness), which can be attained by designing the learning process accordingly. We study this challenging problem in **critical domains with risky states**, where we aim to design a learning mechanism, which is *robust* against random or adversarial perturbations in action execution.

(3) modelling agents Lastly, it is important to discuss the challenge of modelling complex agents, where we study adversarial intelligent agents who cannot be fully observed. We address this obstacle of multi-agent learning deployment in **spatial security domains** demonstrated in the illegal rhino poaching domain.

We summarise the problem statement of this thesis as follows:

Problem Statement: *Learning optimal multi-agent policies and modelling of multi-agent interactions in complex domains is hindered by critical obstacles that we identify as: (1) how to model and design a complex environment and multi-agent interactions effectively, (2) how to design robust learning processes and (3) how to model adversarial agents. Dealing with these limitations will narrow the gap between multi-agent learning theory and practice and thus help tackling real-world issues, for example the space debris removal problem.*

In order to address this problem, following the identified obstacles (1-3), we now define 3 two-fold research questions, which are investigated in this thesis. We first present research

question 1 targeting the challenge of modelling a complex environment and designing a model of agent interaction in the domain of the space debris removal problem. We then propose research question 2 aiming at achieving robust learning process against perturbations in critical systems with risky states. Finally, in research question 3 we focus on modelling complex adversarial agents and finding safe behaviour in spatial security domains.

Thus, we define the first research question, which we address in Chapter 4.

Question 1.A: *How can we model a complex environment such as the space debris removal problem from a game-theoretic and learning perspective in order to understand how agents can optimise their behaviour?*

Depending on the assumptions we make during the design of the model of the complex multi-agent interactions, such as one-shot versus repeated interactions or assuming cooperative versus non-cooperative agents, the quality and the type of a solution might differ in several aspects. It is thus necessary to propose a methodology to compare these different modelling choices. There might exist several metrics describing the quality of the solution such as social welfare, individual utility or fairness. It is interesting to compare these metrics and evaluate how preferring one over the other impacts the environment. We study this problem in the domain of the space debris removal problem, where we discuss the meaning of the different modelling approaches and how important the different solution attributes are in this specific domain. Therefore, this leads to the second part of the first research question.

Question 1.B: *How can we compare various solutions that emerge when following different modelling choices of the agent interaction mechanism and evaluate their effectiveness in complex domains?*

Apart from obstacles discussed above when deploying multi-agent learning methods into real-world applications, we might also need to design a learning process with specific properties such as robustness. Domains with critical risky states such as smart power grids are often sensitive to node destabilisation caused by perturbations such as an attack or a failure, which can have a severe impact on the whole system. Hence, the process of learning effective behaviour needs to be robust to such events to provide safety of the nodes and sustainable use of the whole system. The actors in such critical domains aim to find

effective behaviour profiles, and computing them exactly is not feasible due to the complex and stochastic nature of such multi-agent scenarios. Thus, learning effective behaviour from interaction might be the right approach. This leads to the research question 2, which is answered in Chapter 5.

Question 2.A: *How can we design a robust learning process against random or adversarial perturbations in critical systems with risky states?*

In pursuance of deploying such a robust learning process into real-world application domains there is often a need for theoretical guarantees of proposed methods in order to warrant desired performance. Several convergence proofs of single-agent reinforcement learning methods, mostly based on the theory of stochastic processes, have been proposed [Tsitsiklis, 1994; Jaakkola et al., 1994; Singh et al., 2000]. However, it is not always possible to extend these proofs into other types of learning with for example requirements on robustness. Especially in the field of multi-agent learning any theoretical guarantees are difficult to obtain due to non-stationarity caused by multi-agent interactions. Thus, being able to prove any theoretical properties of multi-agent learning methods is a very active domain of research with little success so far. Therefore, this leads us to the second part of the second research question.

Question 2.B: *To what extent can we guarantee any convergence to an optimal solution when learning policies, which are robust against perturbations in domains with risky states?*

Apart from challenges of multi-agent learning deployment like modelling of complex dynamic environments or designing robust learning processes, which were discussed above and make up the research questions 1-2, we might also face the challenge of modelling of complex adversarial agents. This is the case in spatial security domains, which are defined between two (groups of) agents; the defender and the attacker who move on a graph. The main threat in such problems are adversarial and potentially intelligent attackers who aim to attack some critical targets. The main goal for the defender is to come up with effective behaviour in order to apprehend the attacker and thus mitigate the security threats. Such behaviour could be learned by interacting with the environment and the attacker. An effective approach might consist of learning an opponent behaviour model, which has not yet been studied sufficiently [Albrecht and Stone, 2018]. We therefore define the research

question 3 and address it in Chapter 6.

Question 3.A: *How can we model complex adversarial agents in spatial security domains in order to mitigate the threats they pose?*

Furthermore, such adversarial attackers in complex domains are often not observable or only partially observable. We focus on the spatial security domains, such as the problem of illegal rhino poaching, where the defender gets to occasionally fully observe the location of the adversarial opponent. It is therefore interesting to investigate how such an occasional full observation of the attacker could help in the defender learning process and thus help to mitigate the security threats. We define the second part of the final research question.

Question 3.B: *To what extent can we make use of an occasional full observation of the adversarial attacker in the spatial security domains?*

1.3 Contributions and Thesis Outline

We now present the outline of this thesis, together with highlighting the main contributions. Firstly, we introduce the main theoretical concepts from game theory, reinforcement learning and multi-agent learning used in this thesis in **Chapter 2**. We start with game theory introducing normal-form games, repeated games, and equilibrium solution concepts. We continue with the reinforcement learning paradigm and introduce concepts such as value functions, Bellman equations and temporal difference learning, describing the Markov decision processes and the stochastic game framework. Finally, we introduce multi-agent learning, which can be thought of as an intersection of the two fields of game theory and reinforcement learning.

In **Chapter 3** we describe the application domains together with the methodology we use to study them, where we focus on the multi-agent learning perspective. We discuss three security and sustainability problems, representing real-world domains which are analysed in this thesis: (i) the space debris removal problem, (ii) learning robust policies in critical systems with risky states and (iii) modelling adversarial agents in spatial security domains.

Chapter 4 addresses the challenges of modelling a complex environment such as the space debris removal problem and designing models of agent interactions. Firstly, we present

a high-fidelity space debris simulator and a surrogate model based on the simulator, allowing us to evaluate different removal strategies and their future impact on the space debris environment. We then propose several models of agent interaction from a one-shot model, analysing its Nash equilibria, to a dynamic model, showing how to compute or learn an effective behaviour. Moreover, we discuss several models with different levels of cooperation between agents. Finally, we thoroughly compare various solution types emerging from following different modelling assumptions in the space debris removal problem and evaluate their fairness and their costs in terms of *price of anarchy*, providing a rich recommendation tool for future decision makers.

In **Chapter 5** we propose a novel approach to designing a robust learning process, leading to learning safe and robust policies in critical domains with risky states such as smart power grids. We present a novel temporal difference learning operator which can implicitly deal with rare but significant events potentially corrupting some parts of the system. The presented temporal difference operator can learn safe and robust policies even before such significant events occur in both single- and multi-agent settings. Furthermore we prove convergence properties of the proposed operator.

Chapter 6 discusses the problem of modelling adversarial agents in spatial security domains, with the example problem of illegal rhino poaching. We discuss how partial observability of adversarial opponents can be used to improve learning of an effective behaviour. We present a multi-agent model, based on the framework of stochastic games, which explicitly deals with partial observability of the opponents' location. By combining reinforcement learning with Bayesian inference we propose a novel approach to learning a model of the opponent behaviour which we then use to derive effective strategies. Such strategies are able to more efficiently mitigate the security threats posed by adversaries in spatial security problems.

We conclude this thesis in **Chapter 7** where we answer the research questions posed in Section 1.2 together with the main contributions. Finally, we discuss limitations of this thesis and suggest possible extensions and directions for future work.

2

Preliminaries

This thesis studies the multi-agent learning paradigm in the security and sustainability domains. Before we dive into the main body of this thesis, we present the necessary background. Hence, this chapter describes several theoretical concepts used in this thesis. Topics that are prominent in this thesis are game theory and reinforcement learning. These fields have a common theme: interaction of agent(s) with an environment and potentially with each other. It is worth mentioning there are several types of notation used across the fields of game theory, (multi-agent) reinforcement learning and the closely related problem of multi-armed bandit. In this thesis we use the terms agent/player, reward/payoff/utility, policy/strategy, game/environment interchangeably, depending on the context.

To outline this chapter, we start with describing the main concepts from game-theory. We continue with reinforcement learning, describing Markov decision processes and value functions, arriving to a crucial concept for this thesis - temporal difference learning. Finally, we present multi-agent learning, building on the presented concepts from game theory and reinforcement learning. We now provide a high-level overview of the discussed fields, describing their similarities and differences with the goal of linking them.

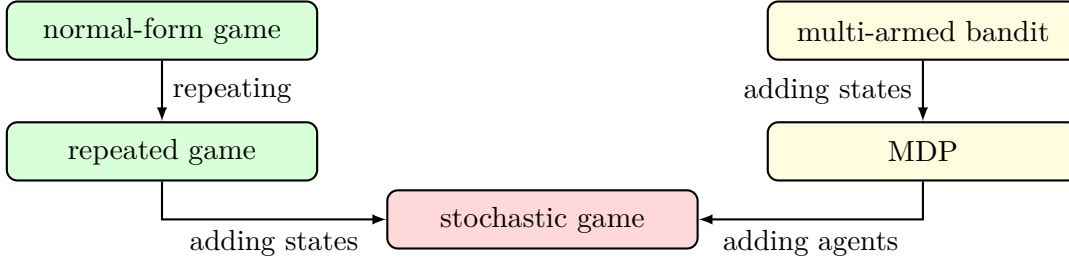


Figure 2.1: Relation between models of agent(s) interaction with an environment. In green are multi-agent models with a single state (stage), in yellow are single-agent models and in red is their generalisation: multi-agent model with multiple states.

Models of agent(s) interaction with an environment In this thesis we consider several models of agent(s) interaction with an environment; single- and multi-agent models and models considering either a single stage or multiple environmental states. In Figure 2.1 we show the relation among several such models in a simplified way to get a high-level overview, omitting some details. A simple one-shot interaction among multiple agents can be modelled as a normal-form game. Considering repeated interactions in the same stage game we arrive at a model of a repeated game. These models (in Figure 2.1 shown in green) assume a single stage (state), hence the action space is the same in any time step and the reward function depends only on the action. Generalising from this to a multi-state interaction, we can use the model of a stochastic game, where the reward function depends now on an action and a state. Furthermore we need to define the state transition function describing how the environment transitions between states by taking actions. On the right side of Figure 2.1 shown in yellow we have a single-agent perspective, starting from the model of a multi-armed bandit, which is a stateless single-agent problem with only actions and rewards. Adding states we arrive at a Markov decision process (MDP), which is then defined over states and actions with a reward function and a state transition function. Considering multiple agents, we move from an MDP to the model of a stochastic game. In this chapter we describe these frameworks in detail. We start with normal-form games, then we formally present Markov decision processes and finally by linking them all together we introduce stochastic games.

		Player B	
		cooperate	defect
Player A	cooperate	-1, -1	-3, 0
	defect	0, -3	-2, -2

Table 2.1: Prisoner's dilemma.

2.1 Game Theory

Game theory studies the interaction between rational agents and aims to find optimal behaviour for such agents. The field of game theory was mainly developed in the 1940s and 1950s but its roots date back to the 19th century and the field of economics. Of several prominent scientists who were at the birth of modern game theory we mention John von Neumann and John Nash. In their seminal works [Von Neumann and Morgenstern, 1944; Nash, 1951] they laid the foundations of two player zero-sum game and the concept of (Nash) equilibria, respectively, providing very important theoretical groundwork. One of the important building blocks of game-theory is the normal-form game, which we now describe.

2.1.1 Normal-Form Games

A crucial concept in game theory is a normal-form game, also known as a matrix or strategic-form game, capturing an interaction between players [Leyton-Brown and Shoham, 2008]. In Table 2.1 we show a simple instance of a normal-form game which constitutes one of the most famous examples of game theory - the prisoner's dilemma. We can see there are two players, player A (the row player) and player B (the column player). Each player can decide between two actions, either to *cooperate* or to *defect*. Depending on the combination of actions the players choose, each player receives a reward. For example, if player A cooperates and player B defects, they receive the reward of -3 and 0, respectively. We now formally define a normal-form game as:

Definition 2.1.1. Normal-form game

A finite normal-form game is defined by a tuple (n, A, R) where

- n is a finite set of players,
- $A = A_1 \times \cdots \times A_n$, where A_i is a finite set of actions available to player $i \in n$, each action $a_i \in A_i$ and each player has $|A_i|$ actions available, a joint action \mathbf{a} is a vector

of actions taken by all the players,¹

- $R = R_1 \times \dots \times R_n$, where $R_i : A \rightarrow \mathbb{R}$ is the reward function for player $i \in n$, assigning a reward r_i .²

Zero-sum games A common type of a two-player game is a zero-sum game, which is a special case of a constant-sum game. In such games the sum of rewards of the two players is equal to some constant c as $r_1(a_1) + r_2(a_2) = c$, $\forall a_1, a_2 \in A_1, A_2$; for a zero-sum game $c = 0$, in other words the gain of one player is equal to the loss of the other player. Such games can be seen as purely competitive, because maximisation of rewards of one player means minimisation of rewards of the other player. In zero sum-games we often use the term *opponent* to denote the other player.

General-sum games Generalizing from zero-sum games we arrive at a broader class of games, where there is no restriction on the rewards. An example of a general-sum game is the prisoner's dilemma shown in Table 2.1, where both players receive bad rewards when both *defect*.

2.1.2 Solution Concepts

Strategy types The main and often only goal of a player is to maximise his reward. The acquired reward depends on action(s) chosen according to a strategy (policy). A player forms a strategy π , which prescribes which actions to take from his action set. A strategy π is then defined as a probability vector, assigning probability of choosing each action. In game theory we differ between two types of strategies: a *pure strategy* and a *mixed strategy*. A pure strategy is a deterministic strategy, prescribing to play one single action, thus assigning probability of 1 to choosing that action from the action set. A mixed strategy is defined by a vector of probabilities describing the probability of choosing each pure strategy i.e., each individual action, thus a player chooses actions stochastically. We say a player is randomising over a set of actions when playing a mixed strategy. All the actions played with a non-zero probability form the *support* of a mixed strategy. A set of all players'

¹In this thesis we use the common notation \mathbf{a}_{-i} to denote the joint action of all agents except agent i , i.e., $\mathbf{a}_{-i} = (a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$.

²Note that we use a notation from reinforcement learning and not the common notation used in game theory, e.g., u_i for the reward (payoff, utility) function, in order to keep consistency throughout this thesis.

strategies form a *strategy profile* $\pi = (\pi_1, \dots, \pi_n)$. We can then define the expected reward for a strategy profile π and a player i as:

$$r_i(\pi) = \sum_{a_i \in A_i} \pi(a_i, \mathbf{a}_{-i}) r_i(a_i, \mathbf{a}_{-i})$$

After defining the basic concepts of game theory, one wants to know what is the solution of such games or how to play in order to maximise the reward. There might exist other requirements for a solution, but the main one is optimality, where the players prefer gaining maximal rewards. Traditionally, the assumption on seeking maximal rewards is based on the concept of *rationality*; out of two possible actions each player always chooses the one rendering a higher reward, which is a standard assumption in game theory (not the case in evolutionary game theory, which is discussed below). However, the concept of optimality of rewards is dependent on other players actions, therefore we need other requirements on the solution. Another such a requirement for a game-theoretic solution is the situation where no player can unilaterally change his strategy to increase his reward. We now present the core concept of game theory - Nash equilibrium.

Nash Equilibrium

The concept of an equilibrium is based on the definition of the *best response*, which can be defined as an action a_i^* where $r_i(a_i^*, \mathbf{a}_{-i}) \geq r_i(a_i, \mathbf{a}_{-i})$ for any action $a_i \in A_i$. We can define the best response more generally for mixed strategies, where a policy π_i^* is the best response if $r_i(\pi_i^*, \pi_{-i}) \geq r_i(\pi_i, \pi_{-i})$ for any mixed strategy π_i . We can now formally present the Nash equilibrium (NE):

Definition 2.1.2. *Nash equilibrium*

A strategy profile $\pi = (\pi_1, \dots, \pi_n)$ is a Nash equilibrium if every strategy π_i is the best response to π_{-i} .

A Nash equilibrium can be explained as a situation where no player can gain by unilaterally changing his strategy. The prisoner's dilemma game shown in Table 2.1 has only one Nash equilibrium which is *defect, defect* even though such strategy yields a worse outcome in terms of sum of rewards (i.e., social welfare - explained below) than *cooperate, cooperate*. Both players cooperating is an unstable strategy, where each player gains if he unilaterally changes his strategy to *defect*. We now present a very important theorem, which gives us a powerful way to reason about games.

Theorem 2.1.1. *On existence of Nash equilibrium [Nash, 1950]*

Every game with a finite number of players and finite action sets has at least one Nash equilibrium in pure or mixed strategies.

In some games there can exist several NE, which can be pure or mixed. This can bring difficulties in deciding which equilibrium to choose, known as the *selection problem*. There are different properties of equilibria which can help us to decide which equilibrium to prefer, for example the level of *social welfare*.

Social welfare In some cases instead of focusing on individual rewards, it is interesting to look at *social welfare* ω , which is a sum of rewards of all the players. Maximising social welfare can be one of the goals in finding optimal strategies, attaining the *social optimum*. Especially in cooperative domains the level of social welfare is the main indicator of quality of the actions chosen by the players. Measuring the quality of equilibria in terms of social welfare can be then evaluated in terms of *price of anarchy*.

Price of anarchy The concept of *price of anarchy* (PoA), introduced by Koutsoupias and Papadimitriou [1999], measures the level of efficiency lost due to selfish behaviour of the players. It compares social welfares ω of two strategies; for example we can compare the worst Nash equilibrium and a strategy π yielding the maximum social welfare out of all possible strategies in a strategy space Π as:

$$PoA = \frac{\max_{\pi \in \Pi} \omega(\pi)}{\min_{\pi \in NE} \omega(\pi)},$$

where PoA is always greater or equal to 1 and the closer to 1 the more efficient solution we get, i.e., the price of anarchy is low.

Multiple NE can also differ in how stable they are against perturbations, which is studied by evolutionary game theory; an (evolutionary) stable equilibrium is stable against small perturbations or deviations in players' strategies, which we discuss in the next section.

2.1.3 Evolutionary Game Theory

There are limitations in studying Nash equilibria only from a static perspective, mainly caused by the assumption of strict rationality, which is often violated in real-world applications, where people's decisions are not always fully rational [Tuyls and Nowé, 2005].

Evolutionary game theory (EGT), originally applied to biology by Smith and Price [1973], tackles some of these limitations and provides more insights into evolution and stability of equilibrium strategies. The analysis of evolutionary dynamics of competing strategies is a powerful way to study the strategic properties of a game. Evolutionary game theory represents a player's strategy by a population of individuals, each of a certain type which corresponds to one of the player's possible actions [Weibull, 1997]. The players are no longer assumed to be purely rational and might not be able to fully observe the game and derive the best response. EGT assumes that players are repeatedly uniformly sampled from the whole population, representing possible perturbations in decision making, showing the whole process of attaining equilibrium while not always being strictly rational [Bloembergen et al., 2015]. A population is defined by the vector $\mathbf{x} = (x_1, x_2, \dots, x_m)$, where $x_i \in [0, 1] \forall i$, and $\sum_i x_i = 1$. The fraction of the population belonging to each of the m types indicates the probability with which a player will play the corresponding pure action from the action space of size m . The *replicator dynamics* defines how the fraction x_i of each type i in the population \mathbf{x} changes over time (\dot{x} denotes derivation over time) due to evolutionary pressure:

$$\dot{x}_i = x_i[f_i(\mathbf{x}) - \bar{f}(\mathbf{x})]$$

where $f_i(\mathbf{x})$ is the fitness (i.e., expected payoff/reward) to type (action) i in the population, and $\bar{f}(\mathbf{x}) = \sum_j^m x_j f_j(\mathbf{x})$ is the weighted average fitness of the whole population. Under the replicator dynamics, the number of types that do better than average will increase, whereas types that do worse will decline. Such a process can generate *evolutionary stable strategies*.

Evolutionary stable strategy Evolutionary game theory studies a refinement of Nash equilibrium, reflecting the dynamic evolution of the equilibrium, called evolutionary stable strategy (ESS). ESS is robust against evolutionary pressure from any new mutant strategy derived from the population, i.e., robust against small perturbations in players' strategies. It is important to note that ESS does not always exist, for example there is no ESS in the game of rock-paper-scissors. Interestingly, every ESS is also a NE but not every NE is also an ESS.

2.1.4 Multi-Stage Games

In this section we describe several types of multi-stage games relevant to this thesis, which are generalisations of normal-form games. We start with a natural generalisation of normal-

form games called *repeated games* (RG), assuming repeated interactions of the same base game, we then discuss *extensive-form games* (EFG) as a generalisation of the normal-form game to multiple stages. Lastly we discuss *Stackelberg games*, representing a special case of asymmetry in players' knowledge about each other. All these games can be finite or infinite, in this thesis we focus on the finite versions.

Repeated games A repeated game is a type of a game, where one same stage-game is repeated over multiple rounds. When deciding on an action players need to take into account the future impact of their decisions on other players behaviour. Therefore, the players consider the history of other players' past plays to form a strategy. For example this is the case in a repeated version of the classic prisoner's dilemma from Table 2.1 called *the iterated prisoner's dilemma*. In iterated prisoner's dilemma, if the number of rounds is unknown, the players might prefer to cooperate i.e., the socially optimal strategy, instead of playing the Nash equilibrium - to defect.

Extensive-form games One can also assume games with dynamically changing stages, which requires a more general framework of extensive-form games (EFG). A game in extensive form is usually modelled using a game tree, which allows to capture the repeated interaction between agents together with potentially incomplete or imperfect information about some parts of the game. In an incomplete information game a player is uncertain about the rewards or the type of the opponent. In the case of imperfect information game a player cannot observe moves made by the other player and needs to reason about several possible decision nodes forming an *information set*. The game of chess is an example of a perfect information game, where all the moves of both players are fully observable. The relation between EFG and normal-form games (NFG) is that every game in extensive form has a unique normal-form representation, however a normal-form game has very often multiple extensive-form representations [Leyton-Brown and Shoham, 2008]. The solution concept for EFG also needs to be generalised, depending on every stage, which is for example *Markov perfect equilibrium*, which is beyond the scope of this thesis, for further reading see Maskin and Tirole [2001]. We discuss a more general but related concept of stochastic games later in Section 2.3.

Stackelberg games Another direction of generalisation from standard normal-form games is considering different abilities of players to observe each other. This thesis focuses

on security domains where there often exists an asymmetry in observation capabilities between the players. Such asymmetry is interesting to study from a game-theoretic perspective, thus we present the framework of Stackelberg games, based on the Stackelberg leadership model, which was proposed in the field of economics for competing firms. In standard normal-form games we assume that the players are taking actions simultaneously and have no knowledge about the strategies of other players. In Stackelberg games we assume that an opponent can observe the agent's strategy before forming his own, by for example observing and reasoning about the past actions of the agent.

A Stackelberg game is defined for two types of players; the *leader* and the *follower*. The leader first commits to a strategy, then the follower observes it and acts upon it. This assumption brings new dynamics into players' decision making and alter the concept of Nash equilibrium. Therefore, in Stackelberg games we seek a refinement of the Nash equilibrium called the *Stackelberg equilibrium* [Stackelberg, 2010].

In recent years the theory of Stackelberg games has been applied to security domains, defined as Stackelberg security games [Kiekintveld et al., 2009]. In many security domains we often face such an information asymmetry, where a potential attacker (follower) might have an access to extra information about the defender (leader) strategy. We call such an attacker the *Stackelberg attacker*; this model of an attacker can be often thought of as the worst-case-scenario attacker. In Stackelberg security games it is often beneficial to find a *Strong Stackelberg equilibrium* [Kiekintveld et al., 2009], which exists in all Stackelberg games and is defined as a pair of strategies where

1. the leader plays a best-response, then
2. the follower plays a best-response, where
3. the follower breaks ties optimally for the leader.

The framework of Stackelberg games possesses several useful theoretical properties, which have been made use of in designing safe policies for real-world security applications, for example the uniqueness of Stackelberg equilibrium [Korzhyk et al., 2011], avoiding the selection problem of Nash equilibria. Security game models have seen several successful deployments tackling real-world security threats; for example the ARMOR system for airport security [Pita et al., 2008], the IRIS tool for scheduling Federal Air Marshals [Tsai et al., 2009] or the PROTECT system for scheduling Coast Guard [Shieh et al., 2012].

2.2 Reinforcement Learning

The field of reinforcement learning (RL) studies how agents can learn from trial and error by interacting with an environment. Reinforcement learning is inspired by learning in humans and animals who, based on an external signal (e.g., pain, pleasure), learn in complex environments on a trial-error basis [Lake et al., 2017]. In RL this external signal is represented by a *reward* [Sutton and Barto, 1998]. The main goal of reinforcement learning is to find actions which maximise a cumulative, potentially discounted, reward. RL thus studies the process of attaining optimal decision making. A very similar problem has been studied across many fields with different names; in engineering: optimal control, in applied mathematics: operational research, in neuroscience: dopamine-based learning, in psychology: animal and human reward-based behaviour or in economics: utility theory together with game theory and the quest for finding optimal decision making.

The reinforcement learning paradigm is often classified as a subfield of machine learning together with supervised and unsupervised learning. In supervised learning we assume having an access to labels of the input data, whereas in unsupervised learning we do not have any such labels, thus the task is to discover hidden similarities among the input data. RL can be thought of as being in between these two approaches, where the core difference is the reward signal, which can be thought of as a relative target label, telling us how good or bad the chosen action (input) is. Thus, the difference is that we do not have the labels (target variables) explicitly but need to approximate them by interacting with an environment.

In reward-based systems we face several challenges like *delayed reward* and *credit assignment*. The problem of delayed reward is the situation where the effect of an action is expressed only many time steps after the action was taken. A related problem is the problem of credit assignment, where due to a delayed reward we might not be sure which reward was caused by which action and thus we might be unable to assign a correct credit (value) to that action. RL is trying to effectively deal with these issues, e.g., it can discount future rewards by using a discount factor γ . The credit is assigned by propagating the reward through a Markov decision process, for example by using bootstrapping, which we will discuss in detail below.

Reinforcement learning algorithms can be divided into two groups; model-free and model-based algorithms. RL is in principle a model-free approach, where we do not know or do not learn the model dynamics, expressed by the state transition function and the reward

function. A different perspective in RL is a model-based approach, where we either assume we know some parts of the model of environment dynamics a priori or we can learn them from interactions with an environment. A model-free approach makes very few assumptions about the environment, mostly only about the reward structure and can be effective for learning complex policies. A disadvantage of a model-free approach is that it requires a lot of experience and is thus in principle slower than model-based approach. A model-based approach gathers knowledge more easily due to partial self-supervision and makes it easier to transfer the learnt knowledge over similar tasks. A disadvantage of a model-based approach is that it potentially needs many assumptions provided by the operator and can be sensitive to wrong assumptions, causing bad performance. In this thesis we consider both of these RL approaches in several security and sustainability domains.

We now discuss a crucial problem of reinforcement learning and other related fields - the exploration vs. exploitation dilemma, which is also studied in the multi-armed bandit problem, which we briefly describe. In this section we then formally introduce the framework of Markov decision processes, which is often used for reinforcement learning. The remainder of this section focuses on core concepts of RL such as policy and value functions or temporal difference learning, lastly we discuss a generalisation of an MDP into partially observable domains called the partially observable Markov decision process (POMDP).

Exploration vs. Exploitation A crucial problem in reinforcement learning is the trade-off between exploration and exploitation. We face a dilemma whether we should exploit the best option observed so far or whether we should explore the state space further with potentially discovering a better option. Imagine you are deciding to which restaurant to go for dinner, you can either pick the best of those you have tried so far or you might try a new restaurant, which might be potentially better than those you already tried, facing the dilemma between exploitation of the best option so far and exploration of new yet unobserved options. There are several proposed ways in reinforcement learning which aim to tackle this dilemma effectively. One of the simplest is ϵ -greedy policy, which explores with probability ϵ and exploits the best option so far with probability $1 - \epsilon$. There exist approaches which balance the trade-off between exploration and exploitation implicitly like Bayesian reinforcement learning [Ghavamzadeh et al., 2015]. This problem is also the main focus of the multi-armed bandit problem, which we discuss next.

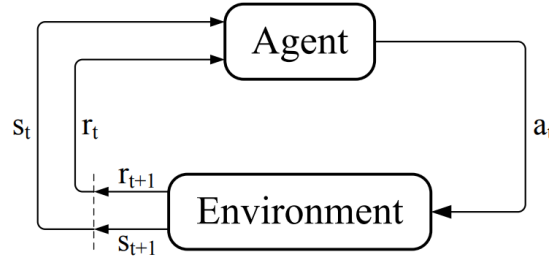


Figure 2.2: Markov decision process: The agent interaction with an environment. Taken from Sutton and Barto [1998].

Multi-armed bandit A closely related topic to RL is the multi-armed bandit (MAB) problem, which can be seen as a special case of RL with only a single state. The multi-armed bandit model studies a classic exploration-exploitation dilemma, where we assume K arms (actions), each having an unknown distribution of rewards. The goal is to find the optimal arm, which maximises the reward. There have been several learning algorithms proposed for MAB, for example UCB, or its adversarial version EXP3 [Auer et al., 1995]. Multi-armed bandit is mentioned here due to a close relation to RL and its importance for studying exploration vs. exploitation dilemma. For further reading see Bubeck and Cesa-Bianchi [2012], who discuss other variants of MAB like contextual MAB or adversarial MAB, which has also been studied in security applications [Klima et al., 2015].

2.2.1 Markov Decision Process

In this section we describe a finite Markov decision process (MDP), which is a key concept for describing the interaction of an agent with an environment [Puterman, 1994]. In Figure 2.2 we show a standard scheme of such interaction; the agent chooses an action a_t at time step t , which then gets executed by the environment, the agent receives a reward r_{t+1} and the environment transitions to the next state s_{t+1} . We now provide a formal definition of an MDP:

Definition 2.2.1. Markov decision process (MDP)

A finite Markov decision process is defined by a tuple (S, A, R, T) where

- S is a finite set of system states
- A is a finite set of actions

- $R(s, a) \in \mathbb{R}$ is the reward function for state $s \in S$ and action $a \in A$
- $T(s, a) \rightarrow s'$ is the transition function, defining the probability of ending in state s' after taking action a in state s .

A system state s encodes all the available information about the current system setting. The action set represents available actions (potentially as a function of a state), the reward function defines a real-valued reward for a state and an action and transition function returns the probability of the system transitioning to a next state s' for a given state s and action a . An important property of a system modelled as an MDP is the *Markov property*, which describes the situation where the current state is fully descriptive (independent) of the past history of states and actions, thus we can write the probability of the system transitioning to a state s_{t+1} is $p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_1, a_1)$. The Markov property is often a necessary condition for convergence and stability guarantees of learning algorithms.

2.2.2 Policy & Value Function

The main goal of RL is to learn an optimal *policy*. A policy π is defined as a mapping from system states to probabilities of selecting actions, and thus prescribes a strategy to an agent in each state, we write $\pi(a|s)$ for the probability of choosing an action a in a state s . Pivotal in finding optimal policies is the approximation of the *value function*, which describes how good a particular state is or how good it is to select a particular action in a given state. Thus, we talk either about a state value function denoted as $V(s)$ or a state-action value function denoted as $Q(s, a)$ for a given state s and an action a . The value or the goodness of a state or a state-action pair is based on the expected *return*, which is a function of current and future rewards, for example a sum. The future rewards are often discounted in order to put more emphasis on the more recent rewards and in case of continuous (non-episodic) task to guarantee a finite sum of the rewards. We denote a discount factor as $\gamma \in (0, 1)$. We can now present formally the state value function, which is defined for a given policy π as:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^T \gamma^t r_t \mid s \right], \quad (2.1)$$

assuming a finite episode of length T . And the state-action value function is defined as:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^T \gamma^t r_t \mid s, a \right]. \quad (2.2)$$

We can now define the optimal value function, using an optimal policy π^* as:

$$V^*(s) = \max_{\pi} V^\pi(s) \quad \text{or} \quad Q^*(s, a) = \max_{\pi} Q^\pi(s, a). \quad (2.3)$$

The process of approximating these functions is based on linking the values between a state and its successor state by using the *Bellman equation*:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} T(s'|s, a) (R(s, a) + \gamma V^\pi(s')), \quad (2.4)$$

similarly for the state-action value function:

$$Q^\pi(s, a) = \sum_{s'} T(s'|s, a) (R(s, a) + \gamma Q^\pi(s', a')). \quad (2.5)$$

We can now use the Bellman equation to find the optimal value function by defining the *Bellman optimality equation* for a state value function:

$$V^*(s) = \max_a \sum_{s'} T(s'|s, a) (R(s, a) + \gamma V^*(s')), \quad (2.6)$$

and for the state-action value function:

$$Q^*(s, a) = \sum_{s'} T(s'|s, a) (R(s, a) + \gamma \max_{a'} Q^*(s', a')), \quad (2.7)$$

The policy according to which we select actions is called the *behaviour policy*, e.g., ϵ -greedy policy. The policy which we use for updating the value function is called the *target policy*. This distinction is important for an on- and off-policy learning, which we will discuss.

2.2.3 Temporal Difference Learning

One of the common types of reinforcement learning is *temporal difference* (TD) learning. TD methods update the state (or state-action) values based on an estimate learnt in previous rounds, which is called *bootstrapping*. Thus, unlike for example Monte Carlo methods the

TD methods do not need to wait for an episode to end to update the values. Temporal difference, also called the *TD error*, is the difference between a new estimate of the value, called the *target* and the current value. A standard TD method state-action value update is then defined for a policy π as:

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha \left[\underbrace{r + \gamma V^\pi(s')}_{\text{target}} - Q^\pi(s, a) \right], \quad (2.8)$$

where α is the learning rate parameter, defining the level of update of the new information to the old one.

Based on the definition of the target, TD methods can be classified as either *on-policy* or *off-policy* methods. In on-policy methods the behaviour policy is the same as the target policy whereas in off-policy methods they differ. We now present several well-known off-policy and on-policy TD learning algorithms, which will be very important for this thesis.

Q-learning A standard temporal difference off-policy learning method is called Q-learning, originally proposed by Watkins [1989], defined as:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_a Q(s', a) - Q(s, a)). \quad (2.9)$$

This function directly approximates the optimal state-action value function which is independent to the followed behaviour policy (e.g., ϵ -greedy).

SARSA A classic on-policy learning method is SARSA, named after the update sequence tuple (s, a, r, s', a') and introduced by Rummery and Niranjan [1994], defined as:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a)). \quad (2.10)$$

The target policy is the same as the behaviour policy. One difference between SARSA and other presented methods is that it needs to wait one more time step in order to update the value function with the succeeding state s' and the succeeding action a' .

Expected SARSA A modification of the on-policy SARSA method was analysed in Van Seijen et al. [2009] called the Expected SARSA, where the Q-value update function

uses the expected value in the next state given the behaviour policy as:

$$Q(s, a) = Q(s, a) + \alpha \left(r + \gamma \left(\sum_{a'} \pi(s', a') Q(s', a') \right) - Q(s, a) \right), \quad (2.11)$$

thus $V(s') = \sum_{a'} \pi(s', a') Q(s', a')$. Expected SARSA is an on-policy method. It can also be seen as a generalisation of Q-learning, where for purely greedy policy π these algorithms are identical because in both $V(s) = \max_a Q(s, a)$.

Tabular methods vs. function approximation In smaller size problems the value function can be represented by a table, e.g., vector or matrix, where the state values or state-action values are stored, respectively. However for larger environments with a high number of states and/or actions, or continuous state or action spaces, this representation is not efficient and computationally very demanding. Thus, it is often preferable to represent the state (-action) value function by function approximation techniques. Such function approximation can be linear (e.g., tile coding) or non-linear (e.g., deep neural network). In this thesis we focus only on tabular methods and leave the use of value function approximation for future work. Nevertheless, in recent years there has been a huge progress in designing RL methods with non-linear function approximation techniques, for single-agent deep reinforcement learning see the survey of Arulkumaran et al. [2017] and for multi-agent deep reinforcement learning see Hernandez-Leal et al. [2018].

2.2.4 Partially Observable Markov Decision Process

A Markov decision process assumes that the agent is able to always fully observe the underlying system state, which might not be often realistic, and thus it is useful to generalise from MDPs to partially observable Markov decision processes (POMDP) [Kaelbling et al., 1998]. The reasons of not being able to observe the underlying state might be for example noise in sensory input, dynamically changing environment or unobservable parts of the studied system. Formally a POMDP is defined by a tuple (S, A, R, T, Ω, O) , where S, A, R, T are defined in the same way as in a classic MDP, but further we define Ω to be a set of observations and O to be the observation probability function. During the interaction the agent receives observations $o \in \Omega$, which he uses to form *beliefs* $b(s)$ about states. The probability of each observation depends on the next state s' and action a , given by the function O as $p(o|s', a)$. In a POMDP the agent cannot observe the underlying state and

needs to maintain a probability distribution over the states according to the observations he made. In other words the belief $b(s)$ about a state s is the probability of the system being in that state. We define the belief update as:

$$b'(s') = \frac{1}{p(o|b, a)} O(o|s', a) \sum_s T(s'|s, a) b(s), \quad (2.12)$$

where $p(o|b, a) = \sum_{s'} O(o|s', a) \sum_s T(s'|s, a) b(s)$ is the normalising constant. This belief update is performed after observing the succeeding state s' , taking an action a and receiving an observation $o \in \Omega$. The belief state is a *sufficient statistic* [Bertsekas, 1995] for choosing optimal actions, because it is a sufficient summary of all relevant past information. Due to this property a system with belief states is equivalent to a fully observable Markov decision process with a continuous state space [Littman et al., 1995].

2.3 Multi-Agent Learning

In this section we formally present the *multi-agent learning* (MAL) paradigm, we build on previously described fields of game theory and reinforcement learning, where multi-agent learning can be seen as their intersection [Nowé et al., 2012]. Multi-agent learning can be described as a process where multiple agents learn to behave so that they achieve their goals, while they interact with other possibly learning agents, who might have similar or different goals (e.g., cooperative or adversarial agents) [Tuyts and Stone, 2018]. In this thesis we mainly focus on multi-agent reinforcement learning (MARL), which is an extension of classic single-agent reinforcement learning to multi-agent scenario. However, apart from MARL there are other approaches to multi-agent learning coming from different fields such as multi-armed bandit, repeated games, extensive-form games or evolutionary game theory [Tuyts and Nowé, 2005], for a survey of these other approaches see for example Hernandez-Leal et al. [2017]. In recent years there has been an immense interest in multi-agent learning with many surveys published, see Panait and Luke [2005]; Busoniu et al. [2008]; Bloembergen et al. [2015]; Hernandez-Leal et al. [2017]; Tuyts and Stone [2018]; Albrecht and Stone [2018]. In this section we will discuss several approaches to multi-agent learning described in these surveys, which are relevant for this thesis. Firstly, we formally define stochastic games, which is a crucial concept for multi-agent reinforcement learning and which we use in this thesis to model the multi-agent interaction.

2.3.1 Stochastic Games

The framework of stochastic games (SG) [Shapley, 1953], also known as Markov games, is a natural extension of Markov decision processes to multiple agents [Bowling and Veloso, 2000], or a generalisation of repeated games to multiple states as shown in Figure 2.1. A finite stochastic game can be defined as:

Definition 2.3.1. Stochastic game (SG)

A finite stochastic game is defined by a tuple $(n, S, A_1 \dots A_n, R_1 \dots R_n, T)$ where

- n is the number of agents,
- S is a finite set of system states,
- $A_1 \dots A_n$ where A_i is a finite set of actions for agent $i \in (1, \dots, n)$, with actions $a_i \in A_i$. We also define joint action $\mathbf{a} = (a_1, \dots, a_n)$,
- $R_1 \dots R_n$ where $R_i(s, \mathbf{a}) \in \mathbb{R}$ is the reward function for agent i , for state $s \in S$ and joint action $\mathbf{a} \in A_1 \cup \dots \cup A_n$,
- $T(s, \mathbf{a}) \rightarrow s'$ is the transition function, defining the probability of reaching state s' after taking joint action \mathbf{a} in state s .

Similarly to Markov decision processes, stochastic games assume the Markov property; the system state fully describes the current system setting and is independent of any past history data preceding the current state. However, this property is often violated in many domains when modelled as a stochastic game, because the agents can not always fully observe the underlying state due to the inability to observe all the other agents influencing the state. The Markov property is crucial for several convergence guarantees in single-agent settings, we cannot thus often rely on those guarantees in multi-agent settings. Especially in real world applications, the Markov property is often broken, which we need to keep in mind when designing multi-agent systems for real-world use. The problem of partial observability of the underlying state in multi-agent settings has been studied, for example using the approach of *profit sharing* as a credit assignment technique based on trial-and-error experience [Arai et al., 2000]. Another way to deal with this problem is to use the framework of *Partially observable stochastic games* (POSG), which explicitly models the uncertainty caused by partial observability of the other agents. POSG is a generalisation of the partially observable Markov decision process (POMDP) to multiple

agents, similarly as a stochastic game is a generalisation of an MDP. In Chapter 6 we investigate an interaction with partially observable adversarial agents, which we model by using the POMDP framework, where we model the opponent behaviour as a part of the transition function and focus mainly on dealing with the partial observability. However, such an interaction could be also modelled with the POSG framework in case of assuming more complex environments, which we leave for future work. For further reading on POSG, see for example Hansen et al. [2004].

2.3.2 Approaches to Multi-Agent Learning

In many real world environments there are multiple agents interacting with each other or influencing each other in some ways. Such interactions can be very complex and very often not fully observable. For such cases single-agent reinforcement learning needs to be extended to *multi-agent reinforcement learning* (MARL). MARL is a substantially more challenging problem than single-agent RL because of the dependencies of agents on each other. What makes learning in a multi-agent environment especially difficult is the *moving target problem* [Tuyls and Weiss, 2012], which describes the situation where agents try to learn dynamically changing strategies of other agents, i.e., learning a continually changing target. This problem is the main source of the non-stationarity of the multi-agent learning. There have been several ways proposed to deal with these complexities, ranging from ignoring the non-stationarity to aiming to estimate the non-stationary behaviour of other agents. We will discuss the most important methods of MAL with respect to the scope of this thesis, where the main focus is on temporal difference learning. Temporal difference learning has been shown to perform well in a single-agent setting, and there have been many efforts to extend Bellman style reinforcement learning techniques to multi-agent settings [Tuyls and Weiss, 2012]. Such approaches have been very successful in zero-sum repeated games, or team repeated games, but less successful in general-sum stochastic games [Shoham et al., 2007], one reason is the ambiguity of defining the goals of the MAL process.

Learning goals in multi-agent learning When designing a multi-agent learning system one needs to set learning goals which we aim to achieve. The main goal of learning is often achieving optimality of behaviour; in many MARL algorithms the main goal is thus to achieve an equilibrium in self-play, which is for example the case for Minimax-Q [Littman,

1994] or under certain conditions Nash-Q [Hu and Wellman, 2004]. Achieving equilibrium in multi-agent settings can be very difficult and often quite restrictive, because there are several problems with NE like non-uniqueness, incompleteness, sub-optimality and rationality, which can be problematic [Shoham et al., 2003, 2007]. Furthermore, Shoham et al. [2003] mention that the link between convergence of NE w.r.t. a game stage and its meaning and performance in dynamic stochastic games is unclear. For example in extensive-form games the concept of NE needs to be extended to a Markov perfect equilibrium. Furthermore, in stochastic games NE might depend on a state and might be very difficult to attain due to the moving target problem and the selection problem, where more NE might exist. Other goals apart from attaining NE have been proposed like stability, for example defined as convergence of an algorithm to a stationary policy [Bowling and Veloso, 2001], Pareto-optimality, social welfare and general fairness [Busoniu et al., 2008], or some other measures like safety or robustness. In this thesis we especially emphasize the need for robustness and safety of any multi-agent learning approach for improving the deployability of such techniques into real-world applications.

Independent learners and joint-action learners When discussing multi-agent reinforcement learning we first need to mention two very simple approaches proposed by Claus and Boutilier [1998], which consider learning among cooperative agents: *independent learners* (IL) and *joint-action learners* (JAL). The authors study these two approaches in terms of classic Q-learning as known in single-agent reinforcement learning, however the general idea of IL and JAL can be applied to other reinforcement learning methods. These two quite general but simple approaches to multi-agent learning are often the stepping stones to build more complex methods in order to achieve desired requirements on the solution. **Independent learners** are assumed to observe only local actions and ignore all the other agents; learning independently by using standard Q-learning. This is a very simple approach to multi-agent learning, where in many applications observing other agents is not possible. Independent learning agents reduce the multi-agent problem to a single-agent one by not considering the other agents and simply perceiving them as stochasticity of the environment. One disadvantage of such approach is that the Markov property is broken and thus we lose most of the guarantees proven for the single-agent case. Nevertheless this naive approach to multi-agent learning can be sometimes quite effective [Busoniu et al., 2008]. On the other hand **joint-action learners** are assumed to be able to fully observe the joint action, i.e., all the agents' actions. While learning in a joint-action manner is more stable and faster

to converge than IL, it is often impossible to attain in real-world applications, where we cannot assume full observability of other agents' actions. One way to tackle this reality gap is to model other agents' strategies in order to estimate the joint action. JAL compared to IL is more memory demanding because the value function uses the joint action as an argument. Thus, the value function is defined on the space of joint actions and suffers from combinatorial explosion in memory requirements depending on the number of agents and the size of their action spaces.

JAL and IL are two extreme cases of multi-agent learning, either fully observing the other agents or totally ignoring their presence, respectively. Hernandez-Leal et al. [2017] propose a more refined classification of multi-agent learning, describing the level of reasoning about other agents; (i) ignore, (ii) forget, (iii) respond to target opponents, (iv) learn opponent models and (v) theory of mind. We now further discuss several joint-action learning methods belonging to (i), (ii) and (iii), which are relevant for this thesis. For the classification of the methods see Figure 5 in Hernandez-Leal et al. [2017].

Other types of joint-action learning Many multi-agent learning methods, in order to approximate the value function, use the joint action, which can be difficult to obtain due to partial observability of other agents' actions. One way to deal with this problem is to estimate the other agents' behaviour and predict their actions. One of the simplest ways to estimate other agents' policies is *fictitious play* (FP) [Fudenberg and Levine, 1998]. FP assumes we can observe past actions of the other agents and form an estimate of their strategies based on the empirical frequencies of their past actions. This approach is based on the assumption that the other agents follow stationary policies (mixed or pure), which ignores the potentially dynamic behaviour. FP is originally defined for single-stage repeated games, however an extension to extensive-form games was proposed [Heinrich et al., 2015].

Another approach to MAL builds on the classic single-agent temporal difference learning algorithm Q-learning. Littman [1994] proposed an extension called Minimax-Q to zero-sum game scenarios, which can be seen as assuming the worst-case attacker, always minimising the Q-value. Moving from zero-sum games to general-sum games, Hu and Wellman [2004] presented Nash-Q, which uses Nash equilibria instead of the minimax operator. Nash-Q has limited applicability caused by the selection problem of NE, where all players need to select the same Nash equilibrium among potentially multiple NE for the algorithm to converge. There have been other extensions of similar approach suggested; Friend-or-Foe [Littman, 2001], Win-or-Learn-Fast (WoLF) [Bowling and Veloso, 2001], Correlated-Q [Greenwald

and Hall, 2003] or “Adapt When Everybody is Stationary, Otherwise Move to Equilibrium” (AWESOME) [Conitzer and Sandholm, 2003].

In the domain of security games a different approach using the concept of the strong Stackelberg equilibrium (SSG) was proposed, known from Stackelberg games and recently widely used in Stackelberg security games [Kiekintveld et al., 2009]. Könönen [2004] combines Q-learning and SSG, making use of the asymmetric learning model with a *leader* and a *follower* as known in Stackelberg games. The solution concept of Stackelberg equilibria overcomes the selection problem of Nash equilibria. However, in such a model both agents need to accept their roles as the leader and the follower and keep a copy of the opponent’s Q function, which is very restrictive and computationally demanding. Using a similar assumption about the asymmetry between players the model of *Bully* and *Godfather* was proposed by Littman and Stone [2001].

3

Methodology & Application Areas

In this chapter we introduce the methodology used in this thesis for applying multi-agent learning and modelling techniques to various security and sustainability domains. By choosing diverse application areas we hope to introduce general and efficient methods and demonstrate the usability of the multi-agent learning approach. Moreover, we explain how to overcome some of the obstacles in deploying the techniques to real-world applications. In this work we focus on three application areas which differ in several aspects but all contain multi-agent interactions and the necessity for security and sustainability. The three areas are:

1. Sustainability of Earth's orbit: The space debris removal problem → **Chapter 4**
2. Safety in critical systems with risky states: Learning robust behaviour → **Chapter 5**
3. Mitigating threats in spatial security domains: Dealing with partially observable attackers → **Chapter 6**

There is a range of assumptions one needs to make about each application area in order to propose a computationally viable but still realistic enough model. In this work we make

use of the game-theoretic framework of normal-form games and stochastic games; where the former is suitable for domains with single-stage interactions between agents and the latter for the domains where the environment evolves with actions taken by agents (see Chapter 2). In simple domains the solution can be often computed exactly by computing an equilibrium strategy or by planning via solving the underlying MDP, however in more complex domains this might not be possible due to computational complexity. Computing a Nash equilibrium becomes quickly intractable [Daskalakis et al., 2009]. In such cases we turn to approximating the solution instead, in our case by reinforcement learning. Learning a solution is often achieved by adapting behaviour with respect to improving the estimate of a value function by repeated interactions with the studied system (described in detail in Section 2.2). Apart from the intractability, the game-theoretic approach also suffers from the selection problem, where multiple Nash equilibria might exist. The learning approach faces another problem, where convergence to a fixed point is often not guaranteed, furthermore the behaviour of the learning process can be very complex and unpredictable [Omidshafiei et al., 2019]. We investigate these issues of both approaches in the studied application domains. We now further present the application domains and discuss the modelling assumptions we make, we compare them and describe their limitations.

3.1 Security and Sustainability in Studied Domains

We now introduce the studied application areas, where all the domains are situated in the security and sustainability context. We present the studied areas in turn as they appear in the chapters later on.

3.1.1 Sustainability of Earth’s Orbit

In Chapter 4 we study the problem of space debris, which is the main threat to sustainability of Earth’s orbit. Since the late 1950s there has been an increase in the number of public and private agencies that have launched a multitude of objects into Earth’s orbits with low or no incentive to remove them after their life span. As a consequence, there are now many inactive objects orbiting Earth, which pose a considerable risk to active spacecraft. The Earth’s orbits are becoming increasingly cluttered with so-called *space debris*, made up by inactive or defunct satellites, rocket bodies, or other parts of spacecraft that have been left behind. NORAD tracks and catalogues objects in orbit, currently listing around

15,000 objects of 10cm² and larger¹. It is believed that the true number of objects is several orders of magnitude larger, with estimates of over 100,000 pieces of untracked debris of sizes 1-10cm² [Carrico et al., 2008]. By far, the highest spatial density of such objects is in the low Earth orbit (LEO) environment, defined as the region of space around Earth within an altitude of 160 km to 2,000 km, in which a large number of active satellites operate. This causes a substantial operational risk, representing defection or even obliteration of spacecraft due to collisions with pieces of debris, which at orbital speeds of approximately 7.5 km/s can cause considerable damage. The density of objects in LEO will most likely increase due to new launches, in-orbit explosions, and the number of object collisions being higher than the capability of the LEO environment to clean itself using the natural orbital decay mechanism. In recent years we have already witnessed two especially severe incidents: (i) a 2007 Chinese anti-satellite missile test producing more than 1,200 catalogued pieces of debris, and an estimated 35,000 pieces of size 1cm and larger, resulting in the most severe orbital debris cloud in history [NASA Orbital Debris Program Office, 2007]; (ii) the collision of the Iridium-33 and Kosmos-2251 satellites in 2009, which was the first accidental hyper-velocity collision of two intact spacecraft [NASA Orbital Debris Program Office, 2009]. This accident produced more than 823 catalogued debris objects, forming two debris clouds in LEO. These incidents introduced a high risk of potential collisions to many active objects in LEO. For example, the International Space Station (ISS) had to perform a manoeuvre in 2011 to avoid a piece of debris from the 2009 Iridium-Kosmos collision [NASA Orbital Debris Program Office, 2011]. One of the earliest analyses of the projected evolution of space debris was done by Donald J. Kessler in 1978 [Kessler and Cour-Palais, 1978; Kessler et al., 2010]. This study led to the definition of the “Kessler Syndrome”, a particular scenario where the density of objects in LEO becomes high enough to cause a cascade of collisions, each producing new debris and eventually saturating the environment, rendering future space missions virtually impossible. Active space debris removal might be the answer to this sustainability threat.

3.1.2 Safety in Critical Systems with Risky States

In Chapter 5 we analyse vulnerabilities of critical systems with risky states, where a destabilisation of individual nodes can have a potentially immense impact on the whole system. We especially study the situation where there might be some random or adversarial

¹See <https://celestrak.com/NORAD/elements/>.

perturbations in action execution in the form of a failure or external attack. Many critical systems exhibit global system dynamics that are highly sensitive to the local performance of individual components. This holds for example for (air) traffic and transport networks, communication networks, security systems, and (smart) energy grids [Cristian et al., 1996; Shooman, 2003; Knight, 2002; Liu et al., 2012]. In each case, the failure of or malicious attack on a small set of nodes may lead to a severe destabilisation. Moreover, innovations in critical systems may introduce additional vulnerabilities to such attacks: e.g., smart power grids communication channels are needed for distributed intelligent energy management strategies, while simultaneously forming a potential target that could compromise safety [Yan et al., 2013]. Our work is motivated precisely by the need for safety in these critical systems, which can be achieved by encoding robustness into the learning process of an effective behaviour against rare but significant deviations, caused by one or more system components failing or being compromised in an attack. An effective approach to preventing severe destabilisation, providing overall security and robustness of the system, is a prerequisite for wide-scale future use of systems with critical states such as smart power grids. Therefore, we do not only require optimality from the sought solution but also robustness against such threats leading to safe performance of our methods. The notion of robustness and safety in reinforcement learning has been studied to some extent in the single-agent scenario [Singh et al., 1994; Morimoto and Doya, 2005; Garcia and Fernández, 2015]. While the inclusion of these properties into multi-agent learning needs to be further investigated [Albrecht and Stone, 2018].

3.1.3 Mitigating Threats in Spatial Security Domains

One of the most pressing sustainability issues caused by the technologically advanced society is its negative impact on animal species, where many species go extinct every year [Thomas et al., 2004]. One important cause of these problems is human activity of poaching, i.e., killing animals to seek profit. This domain represents a classic multi-agent interaction scenario, where we face adversarial agents (poachers). Chapter 6 analyses threats in spatial security domains with two groups of interacting agents with opposite goals. The two groups of players are; the defenders and the attackers. The former represents a group of cooperating agents with the same goal of securing the system, allowing us to model them as a single agent with a joint action space. Similarly the latter consists of a group of potentially intelligent adversaries, with same or similar goal of attacking targets, also modelled as a

single agent, as common in Stackelberg security games (see Section 2.1.4). There are many real-world success stories of modelling some of the security domains as a security game, examples include; the ARMOR system for airport security [Pita et al., 2008], the IRIS tool for scheduling federal air marshals [Tsai et al., 2009], and the PROTECT system for scheduling coast guards [Shieh et al., 2012]. All these works focused on computing an exact game-theoretic solution, which might not be computationally feasible for larger and more complex domains. There has been also research in security domains using the approach of learning; for example the domain of green security games modelling the problem of illegal animal poaching or illegal fishing [Fang et al., 2015] or the border patrol problem [Klima et al., 2014, 2015]. However, none of these works focused on the complex spatial and temporal nature of the problem, which might be necessary to be modelled as a stochastic game. In this thesis we are particularly interested in *spatial* security domains, where the spatial component introduces another level of temporal complexity, where the system can transition among many different states. Here, we assume the agents dynamically move on a graph over varied time periods defined by episode lengths, for which we use the framework of stochastic games. In such complex systems, computing the solution exactly is not feasible due to the difficulty of the problem and thus we approach it by reinforcement learning. Our work is mainly driven by the target application of the illegal rhino poaching problem which is important to study from a sustainability perspective. The main threat here is potential extinction of the whole species. Our goal is to introduce effective security measures to mitigate the rhino killings. We describe our approach on this specific domain, nevertheless, our work is applicable to a wide range of domains containing the spatial component and two groups of agents with opposite goals, guarding or attacking some critical targets.

3.2 Methodology and Problems Classification

We now describe the methodology and modelling assumptions for the analysis of the three application domains, together with discussing the main threats we face in each of the domains. An overview of this thesis is shown in Table 3.1, where we lay out the similarities and differences of the studied domains. In the remainder of the chapter we describe these metrics in detail and provide further context.

chapter	Chapter 4	Chapter 5	Chapter 6
domain	space debris removal	critical domains with risky states	spatial security domains
problem instance	space debris removal	smart power grids	rhino poaching
threat	tragedy of the commons	rare severe attack/failure	adversarial attackers
main challenge	modelling complex environment	designing robust learning process	modelling complex agents
model type	simulator, surrogate	grid world	grid world
learning	model-free	model-free, model-based	model-based
stage/state	single-, multi-	multi-	multi-
agent	single-, multi-	single-, multi-	multi-
framework	NFG, MDP, SG	MDP, SG	MDP, SG
solution	exact(NE), learned(TD)	learned(TD)	learned(TD)
type of contribution	methodology, model, PoA analysis	TD operator κ , $Q(\kappa)$, Exp. SARSA(κ)	approach, BayesRQ
publications	Klima et al. [2016a] Klima et al. [2016b] Klima et al. [2018c]	Klima et al. [2018a] Klima et al. [2018b] Klima et al. [2019]	Klima et al. [2016c] Klima et al. [2018d]

Table 3.1: Thesis main chapters overview: Comparison of modelling choices and assumptions.

3.2.1 Threat Types

In all the studied domains we face various threats and thus there is a need for introducing security measures to mitigate them. They pose risks to the environment or directly to the agents, potentially harming them in some way. If untreated this might cause ineffective behaviour or even damage to the agents or environment. Effective modelling of these threats is thus a prerequisite to optimal behaviour and overall sustainability of the environment. The main threat in the space debris environment comes from the high volume of space debris posing collision risks to active spacecrafts. The threats to the space agencies are then induced by the potential *tragedy of the commons* scenario, where inactivity of the agents can lead to deterioration of the common environment. The space agencies face a dilemma between not acting while waiting for other agents to act or acting by investing into space debris removal. Another type of threat is present in the critical systems with risky states, where we potentially face a rare but significant failure or attack, which can destabilise the whole network and be harmful to all agents. This rare but significant event can be seen as an external control over the system and can have a form of an intentional and adversarial

attack or a random failure. Moreover, the threats in such critical systems are magnified by the sensitivity of individual nodes on the performance of the whole network. We show how such threats in critical systems such as smart energy grid can be faced effectively by learning robust policies. From assuming a rare attack or random failure we move to dealing with a persistent attack in spatial security domains, where we face threats from adversarial attackers, who are strategically attacking critical targets. One example of spatial security domains is the illegal rhino poaching problem, which we model as a security game. The threat in such a problem is represented by poachers killing rhinos, potentially leading to an extinction of the species causing a major environmental impact. The attacks can happen in different locations, representing the spatial component of the problem and bringing further complexity to the model.

3.2.2 Modelling Choices

Based on Chapter 2, which formally introduced the theoretical background of this thesis, we now describe and motivate the use of different modelling concepts to analyse the application domains from security and sustainability perspective.

Single- and Multi-Agent Model Our domains naturally contain multiple interacting agents, therefore the multi-agent learning paradigm is the main framework to consider. However it might be sometimes useful to model these domains as a single-agent scenario as a form of abstraction, which can bring initial insights into the problem. We can either ignore the other agents and use a form of independent learning (see Section 2.3.2). Or in the case of several agents having the same or similar goal, it is often possible to model them as a single-agent with joint action space. We consider these approaches in each of our studied domains. In some settings it is interesting to compare the single-agent solution with the multi-agent one, which gives us deeper understanding of the behavioural dynamics. In the space debris removal problem we analyse this in terms of price of anarchy (Section 2.1.2), evaluating the cost of a decentralised (multi-agent) solution compared to a (single-agent) centralised one. In the case of the critical domains we first consider a single-agent scenario for which we propose a safe and robust behaviour. Then, building on that we introduce multi-agent versions assuming full communication among the agents. The spatial security domains contain two groups of agents, here we model each group of agents with the same goal as a single agent, therefore obtaining a model of a two-agent interaction.

Single- and Multi-Stage Model The studied application domains are naturally multi-stage models, where agents decide on actions over many time steps. However, often the problems can be abstracted and modelled as a single-stage process, with a one-shot interaction among agents. Such model simplifications can bring deeper understanding into the complex interaction dynamics and can represent a higher level of strategic decision making. In the space debris removal problem we first model the interaction among agents as a one-shot single-stage game using the framework of normal-form games, we then consider a dynamic scenario with multiple stages (states) and dynamic strategies modelled within the framework of stochastic games. The multi-stage model is more expressive with more flexibility in agents' behaviour but much more computationally demanding due to potentially large state and strategy spaces. We compare both modelling approaches and point out advantages and disadvantages of each, together with their meaning in the application domain. When studying the safety in critical domains we also consider a multi-stage interaction, modelled as a stochastic game, where the environment can transition between different states. Similarly, in the spatial security domains we use the model of a stochastic game to capture the interactions between the two groups of players with different goals. The framework of stochastic games is more suitable for the studied domains compared to the model of repeated games because the stages dynamically change with respect to the agents' behaviour.

Solution Types By following different assumptions when modelling the studied application domains, we might obtain various solution types, differing in level of effectiveness, robustness or how they are attained. The standard solution in game theory has the form of an equilibrium, where the players are not incentivised to alter their strategies due to not being able to unilaterally improve their payoff (see Section 2.1). In a multi-stage setting the notion of equilibrium becomes somewhat more complex due to the dependence of such equilibrium on each stage, e.g., Markov perfect equilibrium. In multi-stage games like stochastic games, it is often only possible to approximate the solution by learning it, which is often not only less computationally demanding but also more robust against wrong initial assumptions or perturbations in action execution. In the space debris removal problem we consider both approaches, in the single-stage game we compute an equilibrium solution and in the multi-stage game we learn a solution by interaction with the environment, we then extensively compare these two approaches and their respective solution types. From the domain perspective it is important to compare different solutions emerging from following

different modelling assumptions. For example having different goals as expressed by the two different paradigmatic settings: (i) online RL towards individual utility versus (ii) online RL towards social welfare as described by Tuyls and Stone [2018]. We discuss these two different goals in the space debris removal problem and their impact on the future evolution of the environment. When investigating security in critical domains with risky states we focus on the learning approach and provide convergence guarantees of the proposed learning methods. Finally, in the spatial security domains we face high level of uncertainty caused by partial observability of the attacker, which makes computing exact game-theoretic solutions impossible, thus we also focus on the learning approach.

Model-based vs. Model-free Learning We consider two different types of learning: model-free and model-based approach. Reinforcement learning is originally a model-free approach but there has been a lot of work in extending this notion to a model-based type of learning [Polydoros and Nalpantidis, 2017; Kaiser et al., 2019]. In the model-free approach we learn the strategy by interacting with the environment without knowing or learning the dynamics of the environment, represented by the reward and transition functions. On the other hand in the model-based approach we assume we know partly or fully the dynamics of the environment, which can be either learnt from interactions or we might have some a priori information about it. We start in Chapter 4 by considering a simple model-free reinforcement learning algorithm where we only learn value functions of states and actions. Next in Chapter 5 we consider learning, which is model-free with respect to the environment dynamics but model-based with respect to control transitions in the system, defining potential attacks or failures. Finally in Chapter 6 we consider a fully model-based approach, where we approximate the transition function by learning the opponent behaviour, assuming partial observations of state transitions and a priori information about the environment.

3.2.3 Input Data

In many real-world applications and especially those concerning safety obtaining real input data is complicated and often impossible due to sensitivity of such data and the risk of leaking those into the wrong hands. Therefore realistic modelling of these domains is limited in that sense and often it is needed to artificially generate such input data. To fully explore the applicability of multi-agent modelling and learning in various domains, we study both

cases in this thesis; domains with real-world input data and domains where we artificially generate the input data. In Chapter 4 for the space debris removal model we make use of two publicly available datasets: SATCAT and TLE which contain spatial information of objects larger than 10 cm currently in the Earth's orbits. In the two other chapters (Chapter 5 and 6), which concern sensitive physical targets on Earth, we generate the input data artificially with enough parametrisation to allow for wide range of input.

4

Modelling and Learning in the Space Debris Removal Problem

This chapter is based on the following publications:

- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Hennes, D., and Izzo, D. (2016a). Space debris removal: A game theoretic analysis. *Games*, 7(3):20
- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Hennes, D., Izzo, D., Tuyls, K., and Summerer, L. (2016b). Game theoretic analysis of the space debris dilemma. Technical report, ESA Ariadna Study 15/8401
- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Wittig, A., Saper, A., and Izzo, D. (2018c). Space debris removal: Learning to cooperate and the price of anarchy. *Frontiers in Robotics and AI*, 5(54):22

4.1 Space Debris Removal Problem

In this chapter we study the application of the multi-agent modelling and learning paradigm to a complex environment of the space debris removal problem. This chapter presents a new approach to modelling complex environments from a game-theoretic and multi-agent learning perspective. Moreover, we investigate how different models of multi-agent interaction can be effectively designed and evaluated in such a complex domain.

We already introduced and motivated the domain of space debris removal in Section 3.1, we now discuss several ways proposed to deal with the problem. In 2002 debris mitigation guidelines including passive measures were introduced such as the end-of-life management of satellites via de-orbiting or graveyard orbits, which are now implemented in newly launched satellites to counter the space debris risk [Inter-Agency Space Debris Coordination Committee, 2002; Klinkrad et al., 2004]. While effective [Anselmo et al., 2001], it is now widely believed that mitigation alone is not enough to prevent a further build-up of the debris population in low earth orbit (LEO) [Liou and Johnson, 2008; Lewis et al., 2012; Liou et al., 2013]. *Active space debris removal*, though very costly, may offer a solution [Klinkrad and Johnson, 2009; Izzo et al., 2015]. Recently an experimental active space debris removal mission has been performed under the project *RemoveDebris* with a successful in-space test of several debris removal technologies, deployed from ISS [Forshaw et al., 2017]. An active debris removal mission, if successful, has a positive effect (or risk reduction) for all satellites in the same orbital band. This may lead to a strategic dilemma: each stakeholder has an incentive to delay its actions and wait for others to respond. This gives rise to a social dilemma in which the benefits of individual investment are shared by all while the costs are not. This encourages free-riders, who reap the benefits without paying the costs. However, if all involved parties reason this way, the resulting inaction may prove to be far worse for all involved. This is known in the game theory literature as the *tragedy of the commons* [Hardin, 1968]. This dilemma is often studied as a one-shot interaction in which players (the actors) choose their strategy simultaneously and without communication. Most real scenarios however do not follow this abstract set-up, but are rather played out over multiple rounds of interactions, in which previous outcomes may influence future strategy choices.

We start by developing a high-fidelity simulator of the space debris environment in order to project and evaluate the future evolution of the environment depending on different possible actions. These actions represent different debris removal efforts. We briefly present

the simulator in Section 4.2 but leave the main description for Appendix A, where we describe a collision model, a break-up model, an orbital propagator, and future launch scenarios. The simulator while aiming to be realistic is also very computationally demanding. Therefore, using data collected from the full scale simulator, we build and validate an approximate surrogate model that can subsequently be used to efficiently test the effects of various debris removal strategies without requiring the computational power needed to run the full scale simulator. Using the surrogate model, described in Section 4.3.1, we propose different models of agent interaction; we consider static (one-shot) and dynamic (multi-stage) interaction models. Moreover we investigate single- and multi-agent models. In Section 4.3 we therefore design a normal-form game, a Markov decision process and a stochastic game based on the proposed surrogate model. We study how the aggregation of the agents' actions influences the future development of the space debris environment. The objective of this reasoning is to model this strategic dilemma, understand its consequences and compare various centralised and decentralised solution methods. This requires us to provide a way to estimate the effect of certain actions in different time steps on the space environment and assets held by the actors. In Section 4.4 we evaluate and compare solutions obtained from following different assumptions about the agent interaction. We use several evaluation metrics such as *price of anarchy*, which represents the cost of selfish behaviour compared to cooperative behaviour in terms of social welfare. Furthermore, we analyse fairness of different solution types. Finally in Section 4.5, we discuss the obtained results and the implications for potential decision makers.

This chapter is self-contained, however for completeness, in Appendix B we attach our preliminary work, based on a simplified version of the space debris simulator with a simple launch model as described in Appendix A.3. Where we model the space debris problem as a one-shot game, describing the dynamics of the strategic dilemma from a game-theoretic point of view, showing the evolution of equilibria depending on parameters of the model such as the cost of active debris removal. This preliminary study was an important stepping stone to the complex analysis presented in this chapter, which is based on data from improved space debris simulator with a complex launch model, which is fully described in Appendix A.4.

Related Work This study can be placed in the context of several different areas of related work. From a simulation modelling perspective various attempts have been made to accurately predict the evolution of space debris and the resulting risk of collisions for active

spacecraft. Moreover, from a game-theoretic perspective, researchers have utilised similar methods to study related problems of environmental pollution, and the shared exploitation of scarce resources, e.g., greenhouse global warming [Tahvonen, 1994].

Several studies have been published in recent years that consider in detail the effect of active removal strategies to mitigate the space debris problem [Liou and Johnson, 2009; Liou et al., 2010; Liou, 2011]. For example, Liou and Johnson [2009] present a sensitivity analysis on several fixed object removal strategies. They propose removing 5, 10, or 20 objects per year, and compare these mitigation strategies with baselines “business as usual” or “no new launches” and show the effectiveness of object removals. The objects to be removed are chosen according to their mass and collision probability. Our work is inspired by Liou and Johnson’s approach but, in contrast, (i) we consider a multi-agent scenario in which different space agencies independently choose their removal strategy and furthermore (ii) consider a more adaptive scenario in which an optimal strategy for removal can be *learned* based on estimated collision risks and removal costs.

Analysis of complex strategic interactions using game theoretic tools is often hindered by the large action-spaces available to the agents in such scenarios. For example, in the space debris removal dilemma, each possible piece of debris to remove is potentially an action. Additionally, it is often impossible to define payoffs to all (combinations of) actions in advance. This has led recently to the advent of *empirical game theory* [Walsh et al., 2002; Wellman, 2006]. The main idea is to limit the strategy space of each agent by introducing high level generic profiles, or meta-strategies, that capture the main aspects of the interaction. Then, the payoff table for this reduced strategy space can be estimated empirically, either by analysing data from a real system, or by simulating a model of the system. Standard methods and techniques from (evolutionary) game theory can then be applied to the estimated payoff table, e.g., to find approximate equilibria [Jordan et al., 2008]. Such empirical game theoretic analysis has proven valuable in getting insights into various complex real-world domains, such as automated trading [Wellman et al., 2006], auction mechanism design [Phelps et al., 2004], the game of Poker [Ponsen et al., 2009] or Go [Tuyls et al., 2018], collision avoidance in multi-robot systems [Hennes et al., 2013], adaptive cyber-defence strategies [Wellman and Prakash, 2014], and large-scale bargaining [Hennes et al., 2015]. In this work we follow a similar approach but focus on the domain of space debris removal.

The space debris removal dilemma is in many ways similar to other environmental clean-up efforts that have been studied using game-theoretic tools in the past [Bousquet et al.,

1999]. For example, Tahvonen [1994] models carbon dioxide abatement as a differential game, taking into account both abatement costs and environmental damage. More complex models have been studied as well, including for example the ability to negotiate emission contracts [Harstad, 2012]. Another related model is the Great Fish War of Levhari and Mirman [1980]. Although not the same as environmental clean-up, this scenario deals with shared use of a scarce common resource, which potentially leads to the same dilemma in game-theoretic terms, known as the *tragedy of the commons* [Hardin, 1968]. While the problem of space debris can be seen as a tragedy of the commons in this sense, its potential solution by joint effort of different space actors can be modelled as a *public goods game*, in which players jointly need to reach a threshold contribution level in order to produce a public good (clean space, in our setting). A special case of the public goods game, in which contribution of a single player is sufficient, is given by the *volunteer's dilemma* [Diekmann, 1985]. Here, theory dictates that an increase in the number of players decreases the chance of any one player contributing due to the temptation of free-riding, known in psychology literature as the *diffusion of responsibility* [Darley and Latané, 1968]. The space debris removal dilemma presented here is more complex than both game-theoretic models, as we allow for different contribution levels as well as different stakes between the players. Many of the aforementioned studies has focused solely on a (simplified) mathematical model of the underlying system. In contrast, we use a complex simulator to obtain an approximate model which can then be used to study the outcome of various fixed strategies, as well as learn new dynamic strategies that may outperform the fixed ones. In addition, while most previous work treats the dilemma only as a one-shot (or repeated) game, we here propose both, a one-shot game and a more realistic scenario in which different strategy choices can be made at different points in time, which we model within the framework of stochastic games. Furthermore, we discuss the application of reinforcement learning methods to learn efficient strategies in such games.

Recently, there has been work using the learning approach in tragedy of the commons problems, analysing the dynamics of cooperative solutions [Leibo et al., 2017; Perolat et al., 2017]. These works assumed partially observable domains with potentially unknown underlying model whereas in this work we assume fully observable surrogate model known to all the players. Another related work studying cooperation in public goods games uses a version of reinforcement learning called directional learning to (mis)learn and achieve more cooperative outcomes deviating from Nash equilibria [Nax and Perc, 2015]. This method is studied in an evolutionary setting based on one-shot interactions, whereas we study a

more complex stochastic game in which dynamics depend on sequences of actions taken by the players. We also mention the related (interdisciplinary) body of work focusing on the evolution of cooperation in populations of self-interested agents, often modelled using methods from evolutionary biology or statistical physics [Perc et al., 2017]. While those approaches help to better understand why cooperation happens in (human) society on a macro scale, here we focus on the adaptive learning process on the micro-level of individual players. Although parallels can be drawn (see, e.g., Bloembergen et al. [2015]), this type of analysis falls outside the scope of our current study.

Finally, we study the inefficiency of decentralised solutions in the active debris removal problem. The main tool for such analysis is the price of anarchy (PoA), first introduced by Koutsoupias and Papadimitriou [1999], however the study of inefficiency of Nash equilibria is older [Dubey, 1986]. For a general introduction to inefficiency in non-cooperative games we refer the reader to work of Roughgarden and Tardos [2007]. PoA analysis has been used in many domains, to name a few we state selfish traffic routing in a congested network [Roughgarden, 2005] or auctions [Roughgarden et al., 2017]. In our work we focus on a more restricted scenario with PoA evaluation, similar works *measure* PoA [Knight et al., 2017] or analyse division fairness [Aleksandrov et al., 2015].

4.2 Simulating Space Debris Environment

In this section, we briefly introduce a full scale simulator, which we developed to predict the impact of active removal of space debris objects on the future space environment and in particular on the assets of each agent. In order to successfully model the complex interactions and dynamics in the space debris removal problem, we develop a high-fidelity simulator of the space debris environment, which enables us to model and analyse the future evolution of the space debris problem considering potential active debris removal. Apart from the recent in-space testing of active debris removal [Forshaw et al., 2017], no active removal strategies of actual debris have been attempted yet, thus there is only very limited existing data on their cost and effect. Any impact of such action can be thus only simulated. Furthermore, the space environment, similarly to the climate on Earth, only changes over relatively large time scales of many decades. To measure any effect of current actions, it is necessary to simulate at least one century into the future. This of course introduces large uncertainties to the outcome as it requires modelling of human behaviour, i.e., future launch activity, over the next century. Therefore, we consider different launching scenarios.

Predicting the future of space debris is in general a very difficult task and necessarily uncertain, thus we focus on the analysis framework for a wide range of potential future trajectories of space debris development, providing enough generality.

We propose a simulator with two different models of future spacecraft launching. (1) a simple launch model by repeating a past launch sequence, which aims to replicate the simulators in related work (e.g. Liou and Johnson [2009]). And (2) a complex launch model with several different spacecraft classes with non-linear evolution in time, following the mitigation guidelines [Inter-Agency Space Debris Coordination Committee, 2007] such as the end-of-life management of satellites via de-orbiting. We fully describe these two launch models in Appendix A.3 and A.4, respectively. The simple launch model was used for our preliminary work in Appendix B. The complex launch model is used for the main analysis in this chapter.

The simulator is built on top of the Python scientific library PyKEP [Izzo, 2012]. PyKEP provides basic tools for astrodynamics research, including utilities to interface with online databases such as the SATCAT¹ and TLE (two-line element set)² databases, which provide orbital information on all active (not decayed) objects in the low Earth orbit (LEO) regime we are studying, including the orbital elements that uniquely identify an object's orbit, and which are used for orbit propagation. These databases provide the input to our simulator. PyKEP also provides an implementation of the SGP4 satellite orbit propagator (via libsgp4³), which we use extensively in this work. We fully describe the space debris simulator in Appendix A together with all the modules and the two different launch models.

4.3 Models of Agent Interaction

In this section we present several models of agent interaction for the space debris removal problem. Firstly we introduce a surrogate model based on the simulator with the complex launch model, described in Appendix A.4, which allows to model wide range of future evolution of the space debris environment. We present the analysis with the proposed *conservative* launch scenario, however our model is general and can be used with any launch scenario as proposed in Appendix A.4. Using the surrogate model we define a normal-form game, a Markov decision process, which we then extend to form a stochastic game. We

¹<https://celestrak.com>

²<https://www.space-track.org/>

³<https://github.com/dnwrnr/sgp4>

present several solution concepts; comparing single-agent, multi-agent, static (one-shot) and dynamic strategies and thoroughly analyse their effectiveness and outcomes on decision making by using concepts of fairness and price of anarchy (PoA) (formally presented in Section 2.1.2).

4.3.1 Surrogate Model of the Space Debris Simulator

The full scale simulator introduced in Section 4.2 and fully described in Appendix A accurately models the space environment evolution given the complex launch model, but it is unfortunately computationally demanding. In order to facilitate efficient experimentation with different debris removal strategies we design an approximate surrogate model that effectively captures the dynamics of the system but is computationally fast. In this section we describe the intuition and implementation of this surrogate model. We also validate the approximation by comparing its projected dynamics with those given by the full scale simulator in Appendix C.

Implementation

Firstly we introduce the *threshold for removal* of risky objects, which represents *active debris removal*. This is a different approach to previously considered removal efforts, which we investigated in our preliminary work (see Appendix B), where we now better quantify the impact of removal actions. The measure is the expected number of new debris should a collision occur. This new strategy concept can better prioritise the removal of objects causing massive collisions. The first step to building the surrogate model is to run Monte Carlo simulations for different settings of the *threshold for removal* of risky objects. We prevent all collisions that produce an expected number of debris larger than the given *threshold for removal* from happening by removing the risky objects causing these collisions. Figure 4.1 shows the evolution of the total number of objects in orbit for different thresholds for removal and the cumulative number of lost active assets (spacecraft) for the same scenarios. The outcome for every threshold setting in every time step can be evaluated by the gradient of the curve in that time step, which is based on the (expected) number of collisions (defining the number of objects injected into the environment, see Figure 4.1b) and the (expected) number of lost active assets in every time step of the Monte Carlo simulations. We can use these two metrics to define the set of actions and the reward function.

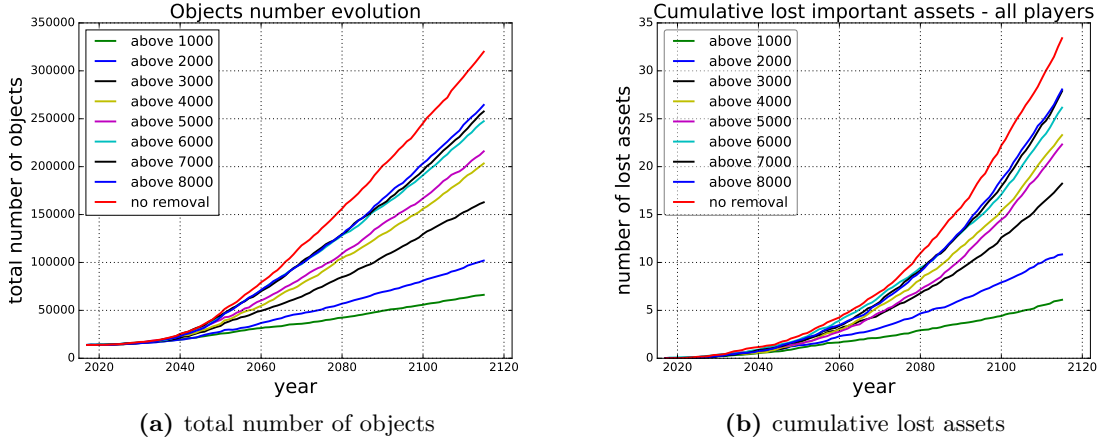


Figure 4.1: Projected evolution of the (a) total number of objects and (b) cumulative lost assets, assuming the complex launch model, in the next 100 years for different removal strategies, e.g. *above 8000* - removing all the objects causing in expectation a collision producing more than 8000 debris pieces.

We can view every point on every curve (system evolution for given threshold) as a potential system state. We restrict states to discrete time steps (decision points for policy change, e.g., 2 years) and assume that the system can transfer between the states by taking removal actions (joint action in the case of multiple actors) defined by given threshold for removal. Due to computational intractability we cannot run simulations for all combinations of (joint) actions and state transitions and hence we propose an approximate surrogate model for evaluating the effect of the removal action on the environment.

This model is based on transitions between the different curves (shown in Figure 4.1) depending on the actions taken. Note that the gradient of each curve is dependent on the total number of objects in the system and the year (which determines the number of active assets). Therefore, we propose *shifting* between curves either (i) horizontally – keeping the same level of total number of objects or (ii) vertically – keeping the same year and therefore the same number of active assets (remember that the number of active assets and their size distribution is only dependent on the launch scenario). This *shifting* between the curves represents transitions between different states of the system. Note that the steeper the curve gets (larger gradient), the higher the risk for collisions will be. If players increase their effort, this means shifting to a lower curve, they move either down or right. A decrease in effort means moving up or left. Intuitively, increasing (decreasing) the effort

should decrease (increase) the gradient. Shifting down when increasing effort (or left when decreasing) gives us a lower bound on the gradient of the curve, as we get an optimistic estimate of the further development of the system by underestimating the total number of objects (or the number of active assets when moving left) in the system. Following the same logic we get an upper bound on the gradient when moving right when increasing effort (or up when decreasing), which can be thought of as a pessimistic estimate where we overestimate the number of active assets (or the number of total objects).

We run 100 Monte Carlo simulations for each scenario to base our surrogate model on. The MC simulations give us for each time step and each threshold for removal the expected number of collisions (and their size in terms of the expected debris resulting from that collision). Basically, the players' actions consist of removing (or deciding not to remove) the difference in expected number of collisions between two curves, and thus moving from the one to the other curve. As discussed, the way in which the system moves between curves can be defined to be either optimistic or pessimistic, to give a lower or upper bound on the expected number of collisions. In the end, this results in a piece-wise combination of the different curves based on the removal actions taken. In Figure 4.1 never removing simply means sticking to the uppermost curve; always removing everything that would produce more than 1,000 debris pieces means sticking to the lowest curve; and any other combination leads to a mixture in between those two extremes.

We validated our surrogate model to well approximate the simulator in Appendix C.

4.3.2 Deterministic Game Model of Space Debris Removal

We base the game model on our surrogate model as described above. We assume the model to be deterministic and think of it as a special case of a stochastic game with deterministic transitions. In this chapter we use the term *stochastic game* due to the common usage in the literature, but *deterministic game* would be more accurate. The game models situations in which multiple players interact. Each player selects an available action given the current state, and the game transitions to a next state based on the combination of actions taken. This makes stochastic games an intuitive framework within which to study strategic decision making between multiple parties, or multiple learning agents. We start by modelling the environment as a deterministic Markov decision process (MDP) (defined in Section 2.2.1), then move to a normal-form game formulation (defined in Section 2.1), and finally we define a stochastic game model (Section 2.3.1). The MDP model assumes a single player scenario.

Building on that, assuming multiple players, we arrive at the stochastic game model.

Markov decision process based on surrogate model

In our model, the underlying environment dynamics are independent of the different players, and governed fully by the sum of actions taken (the chosen threshold for removal). As such, the stochastic game reduces to the special case of a Markov decision process (MDP). In order to transform the validated surrogate model into an MDP, we need to define the state space, the players, their action spaces, the (immediate) reward function, and the state transition function. Each of these will be described in detail below. The general intuition behind the MDP model is the following. At each time step (e.g., per year) players decide how much effort to invest next. This decision can be based on e.g., past actions, the current state of the environment, and budget limitations. The joint effort of all players determines the state evolution of the environment, i.e., the growth rate of the debris population for the coming time step (see model description above). An important question from a design perspective is whether effort will be treated as a discrete or continuous variable. The underlying model (as described above) is inherently discrete, based on a set of thresholds. The reward function can (naturally) be based on the expected number of lost assets between two time steps (states), plus the removal effort. If effort is expressed as an expected number of removals, this means we can get a monetary value by multiplying effort with the cost of removal.

State space In principle the state space is infinite (continuous) along the dimensions of time and current number of objects in the environment (see Figure 4.1). We discretise the state space along both dimensions: 1) by fixing the time steps to for example once every two years, and 2) by fixing the number of allowable states at each time step, uniformly between the top and bottom curves in Figure 4.1. This will fix the total number of states of the MDP.

Action space As our surrogate model is based on a notion of threshold, i.e., not deciding *how many* objects to remove exactly but based only on the *impact* of the potential collision, it seems natural to base the action space on these thresholds as well. Thus, the action space is defined by the discrete thresholds, where for each such a threshold in any given time step we have the expected number of removed objects $\mathbb{E}(n_{rem})$ and the expected number

of lost assets $\mathbb{E}(n_{lost})$ during the next time step. This data comes from the Monte Carlo simulations. The definition of the action space partially defines how the cost associated with each action will be defined. In this case, the cost in terms of future losses is directly given by the effort (threshold) curves in Figure 4.1; however the cost of removals will vary (“everything above a threshold” can be any number of removals, and this will vary over time).

Reward function The (joint) reward function is naturally given by the cost of lost assets plus the cost of removal efforts. Minimising these costs means maximising reward. A reward in the underlying MDP can be also thought of as the environment welfare, which we use in our experimental analysis. We define the reward function $R(s, a)$ for a state s and an action a as a sum of the cost of losing assets and the cost of object removal efforts in the next time interval:

$$R(s, a) = \mathbb{E}(n_{lost}) + \lambda \mathbb{E}(n_{rem}) \quad (4.1)$$

where $\lambda = \frac{C_R}{C_L}$ is a ratio between cost of removal C_R and cost of losing an asset C_L . In this section we do the analysis for different levels of λ , because, similarly as in our preliminary work (Appendix B), we are only interested in the relation between cost of removal and cost of losing an asset and not in their actual values, which can be difficult to determine. $\mathbb{E}(n_{lost})$ is the expected number of lost assets in the next time interval and $\mathbb{E}(n_{rem})$ is the expected number of removed objects in the next time interval defined by the action a in environment state s (given by the threshold curves in the surrogate model).

Transition function The transition function of our MDP is defined by the underlying surrogate model. Given the action a and the current environment state s , the transition function $T(s, a)$ deterministically returns the successor state s' . The function follows the *curve shifting* method explained above in Section 4.3.1.

Stochastic game based on surrogate model

We are now ready to define a game-theoretic model based on stochastic games. We will define the components of stochastic games based on the underlying MDP as described above.

Players We do not consider actual space agencies as we did in our preliminary work, shown in Appendix B, but instead generalise the definition of a player. Players are defined solely by their size, expressed in terms of their number (share) of active assets. This number in turn determines their risk given the current state of the environment. Since we do not discriminate between objects of different players in our model, we assume that the risk scales linearly with the number of assets owned by the players. Each player i has a share ξ_i of assets representing the size of the player. The value of ξ_i is in the range $(0, 1)$ and represents the proportion of player assets to all assets in the environment, thus $\sum_n \xi_i = 1$.

State space The state space is identical to that of the MDP model. Note that due to multiple players the Markov property is broken. Nevertheless we still apply learning methods to derive players strategies. We assume the players can fully observe the underlying state, i.e., the time period and the total number of objects in the system.

Action space Defining the joint effect of several individual actions in the multi-player game is not straightforward. We define a joint action \mathbf{a} as a sum of removal efforts of the players, which is a sum of the expected number of removals for each player's chosen threshold. Then we map this sum to a joint threshold. For this joint threshold we obtain a total expected number of removed objects and total expected number of lost assets from the underlying MDP. We then proportionally divide the expected number of removed objects to each player according to their expressed effort. This method will enable us to define individual rewards.

Reward function We define the reward function based on the MDP reward function definition but now considering the joint action. Thus, the reward function $R(s, \mathbf{a})$ for state s and joint action \mathbf{a} is defined as a sum of the cost of losing assets and the cost of object removal efforts in the next time interval, which is equal to sum of all players' rewards:

$$R(s, \mathbf{a}) = \mathbb{E}(n_{lost}) + \lambda \mathbb{E}(n_{rem}) = \sum_{i \in n} R(s, a_i) = \sum_{i \in n} \left(\mathbb{E}(n_{lost}^i) + \lambda \mathbb{E}(n_{rem}^i) \right) \quad (4.2)$$

$\mathbb{E}(n_{lost}^i)$ is the expected number of lost assets in the next time interval for player i and $\mathbb{E}(n_{rem}^i)$ is the expected number of removed objects in the next time interval for player i .

Individual reward for player i is defined as $R(s, a_i) = \xi_i \mathbb{E}(n_{lost}) + \lambda \mathbb{E}(n_{rem}^i)$, where ξ_i is the share of important assets of player i .

Transition function The transition function is defined according to the underlying deterministic MDP based on the surrogate model. Given the joint action \mathbf{a} and the current environment state s , the transition function $T(s, \mathbf{a})$ deterministically returns the successor state s' . The function follows the *curve shifting* method explained above in Section 4.3.1.

4.3.3 Dynamic Decision Making in the Space Debris Problem

We look at two types of decision making, differing in complexity. Firstly, we focus on the one-shot scenario with *static* strategies, similar to the method we used in our preliminary work in Appendix B, where the players fix their strategies at the beginning of the simulation and stick with those until the end. Secondly, we analyse *dynamic* strategies, where the players can dynamically decide on their action in every time step based on other players' past actions and the development of the environment. Furthermore we analyse and compare a single-agent and multi-agent model. Thus, we can divide the analysis into four types of decision making: (i) single-agent static, (ii) multi-agent static, (iii) single-agent dynamic and (iv) multi-agent dynamic. We discuss these variants in turns.

Single agent

Firstly, we assume only one agent (player) in the system. An entity of n cooperating players can be thought of as a single agent, where every action of the agent is a joint action of the players defined as $\mathbf{a} = f(a_1, \dots, a_n)$, where f is a function aggregating several actions into one joint action. Having only one agent in the system we can directly solve the underlying MDP to find an optimal strategy for the agent. The optimal strategy is given in the form of a sequence of optimal actions across the time horizon T as $\pi^* = (a_1^*, \dots, a_T^*)$. Since the state transition function T is assumed to be deterministic in our surrogate model, applying strategy π^* to the MDP will give us a fixed sequence of states and fixed sum of (discounted) rewards.

We differentiate two levels of complexity for the optimal strategy. Firstly, we consider a static strategy, where one fixed action is repeated for the complete duration of the time horizon, as in our preliminary work, shown in Appendix B. Secondly, we consider a dynamic strategy, which can consist of different actions taken at different time points. In this case

the agent can dynamically change action during the course of the MDP until reaching the final (goal) state. The optimal strategy for the static case can be found by simply maximising the sum of rewards over the strategy space, which has the size of the discrete action space $|A|$. Moving from the static to the dynamic case, the problem of finding the optimal strategy becomes more complex. Now, the strategy space consists of all possible sequences over all discrete actions, which is of size $|A|^T$ for time horizon T . To find the optimal strategy we have to solve the underlying MDP, which can be done by dynamic programming for a small strategy space, or by reinforcement learning for a large strategy space.

Multiple agents

From a single agent scenario we move to a multi-agent scenario. We consider n agents (players), and analyse the interaction among the players over the underlying MDP by using the game model. We assume the players do not cooperate and are self-interested – in the case of cooperation we can model the problem as a single agent scenario as described above. Again, we are interested in finding optimal strategies for the players; however optimality in a multi-agent scenario can be defined in various ways. One way to define optimality is by finding equilibria solutions, another way is by maximising the global welfare. In this section we consider and compare both approaches.

As in the previous section we differentiate between two levels of complexity in the decision making process. Firstly, we look at static strategies, defined as sequences of a repeated fixed actions. In the multi-agent scenario this can be described by a normal-form game and solved by finding Nash equilibria of such game. We are interested mainly in pure equilibria, because mixed strategies are typically difficult to obey and maintain in real-world settings. Especially in the space debris removal, where the removal actions are very costly and the players plan these over several years, it is unlikely to decide on them at random, i.e., choosing mixed strategies, as we also discussed in the preliminary work, shown in Appendix B.3.

Secondly, we analyse the case of dynamic strategies, where the players can take different actions in every time step. The solution is a sequence of actions for each of the players. The definition of optimal strategies in multi-agent dynamic systems might be ambiguous as discussed in Section 2.3.2. Furthermore, the strategy space for n players is large even for a small action space and short time horizon, and grows exponentially in number of actions.

As a result, solving the resulting stochastic game explicitly by for example planning might be impossible or intractable. Thus, the only feasible way to find optimal or near-optimal solutions is to approximate them using e.g., reinforcement learning.

Learning an optimal strategy

In the space debris removal decision making process we face the problem of *delayed reward*, where the effect of immediate actions (object removal or passivity) will fully come into effect only after many years, making reward-based decision-making difficult. Temporal difference methods tackle the delayed reward problem by bootstrapping, i.e., building iteratively more accurate models by incorporating expected future returns into the learnt reward function (see Section 2.2.3 for further explanation of temporal difference learning).

Evaluation metrics

In this section we want to analyse and compare different decision making models. The decision making is based on the underlying MDP built on the surrogate model. Therefore, the main evaluation metrics are based on the concept of reward. The decision making process is prescribed by a given strategy, which can be evaluated in terms of (discounted) rewards both from an individual perspective as well as from a global (environment) perspective. Thus, in the analysis we use the concept of social welfare ω , which is described as the sum of all players' rewards and can be thought of as a global environment outcome.

We use the metric of social welfare to compare the above stated approaches using the concept of price of anarchy (PoA) (see Section 2.1.2), which measures efficiency between two sets of strategies, Π_1 and Π_2 , where the latter is assumed to be worse than the former. PoA compares welfares (sum of rewards) obtained from the different strategies and evaluates the cost of choosing one over the other. PoA can be defined as:

$$PoA = \frac{\max_{\pi \in \Pi_1} \omega(\pi)}{\min_{\pi \in \Pi_2} \omega(\pi)} \quad (4.3)$$

In our experiments the welfare ω is always negative, thus we redefine the PoA so that the obtained values are consistent with standard PoA values, i.e., $PoA \geq 1$, thus defined as:

$$PoA = \frac{\min_{\pi \in \Pi_2} \omega(\pi)}{\max_{\pi \in \Pi_1} \omega(\pi)} \quad (4.4)$$

For strategies from strategy spaces Π_1 and Π_2 , ω is the environmental welfare, which can be also seen as the underlying MDP reward for joint strategy of the players. We extend the classic notion of price of anarchy concept to compare solutions from different models such as varying number of agents forming the strategies or different flexibility assumptions of such strategies. Therefore, the strategy spaces Π_1 and Π_2 are defined based on the given models. We use this metric to compare outcomes from different model designs. In our experimental analysis we use two variants of price of anarchy to measure efficiency, (i) PoA between single-agent and multi-agent as a price for selfish behaviour of the agents and (ii) PoA between static and dynamic strategy as a price for not being able to flexibly react to changes in the environment. We notate the first as PoA_m (single-agent vs. **m**ulti-agent) and the second as PoA_d (static vs **d**ynamic). Therefore PoA_m describes the cost for a selfish decentralised (multi-agent) strategy compared to a cooperative centralised (single-agent) strategy. PoA_d then describes the cost of a static inflexible strategy compared to a dynamic flexible strategy, which can potentially better react to changes in the environment.

We define the concept of *fairness* in the space debris removal game based on player's i share ξ_i as described in Section 4.3.2. *Fairness* is based on the assumption that a level of the removal effort should be proportional to the size of the player. We define *fairness* as $\phi_i = \frac{\omega * \xi_i}{r_i}$, where r_i is a reward for player i . If $\phi_i = 1$ we say that player i behaves fairly, if $\phi_i > 1$ we say that player i behaves positively unfair, meaning he gets a higher reward than he deserves (removing less than what would be fair for his share of assets) and if $\phi_i < 1$ we say he behaves negatively unfair, meaning he gets a lower reward than he deserves (removing more than what would be fair for his share of assets). We also define total fairness as a sum of differences from a fair case for each player i as $\phi = \sum_i |1 - \phi_i|$. The total fairness describes the quality of a solution. If $\phi = 0$ we have a fair solution, the greater the value of ϕ is the less fair solution we have. In Table 4.1 we state a list of notations to help the reader to better orientate in the following sections. In the next section we experimentally compare different scenarios using these evaluation metrics. We perform a thorough analysis for different levels of the ratio λ and shares of assets ξ_i .

4.4 Evaluation of Different Models of Agent Interaction

We base all our experiments on the surrogate model, which is built on the data from Monte Carlo runs of the space debris simulator. Thus, we have the expected number of lost assets $\mathbb{E}(n_{lost})$ and expected number of removed objects $\mathbb{E}(n_{rem})$ for every time step

PoA_m	price of anarchy comparing single-agent and multi-agent scenario
PoA_d	price of anarchy comparing static and dynamic scenario
λ	ratio between cost of removal C_R and cost of losing an asset C_L
ξ_i	share of important assets of player i , i.e., size of player i
ω	welfare, i.e., sum of all players' rewards
ϕ_i	fairness for player i
ϕ	total fairness

Table 4.1: List of notations used in the stochastic game model of the space debris removal.

and every *threshold for removal* (see Figure 4.1). The *threshold for removal* is defining the discrete action space over a set $\{1000, 2000, 3000, 4000, 5000, 6000, 8000, 9000, 10000, NR\}$, where NR is “no debris removal” as defined in Section 4.3.2. The previously described surrogate model can be used to efficiently compare various (predefined) debris removal strategies, investigate the effect of various parameter choices, and to learn an optimal strategy. Our surrogate model is based on *curve shifting* as described in Section 4.3.1, in order to approximate well we restrict the players from abruptly changing the removal thresholds, meaning they can only shift one up, one down or stay at the same threshold level in every time step. The initial choice of action (threshold) is arbitrary. In this section we describe several experiments that highlight each of these possibilities in turn. In all our experiments we assume a 100 year time horizon and decision time step 10 years, so we have 10 decision making points. The time horizon is based on the ISO standard on space debris mitigation [International Organization for Standardization, 2011] agreed to by all space agencies. It sets the time frame for graveyard orbit stability simulations to 100 years. Given the generally quite long lead time for space missions, averaging about 7.5 years for governmental satellite operations [Davis and Filip, 2015], the 10 year decision time step seems a reasonable time for space actors to decide on strategy. Note that decreasing the decision time step to for example 2 years, would accordingly increase the state space with 50 removal decision points in the 100 year time horizon. We also point out that in our model the violation of Markov property is limited for any choice of decision time steps, since we assume that all the agents have access to the simulator and the surrogate model. They can therefore derive the underlying state from the observed year, total number of objects and the surrogate model, prescribing the gradients of expected number of lost assets and expected number of collisions for different removal actions.

As described in a previous section we analyse several scenarios. Our goal is to compare

solution quality of static vs. dynamic scenario and single-agent vs. multi-agent scenario and combinations of these. We first need to describe how we can evaluate these scenarios. We will see that some can be computed and some need to be learned due to the high complexity. We first demonstrate and describe the different scenarios on fixed settings of $\lambda = 0.1$, which is the ratio between cost of removal C_R and cost of losing an asset C_L and, in the multi-agent case, the share of important assets $\xi_A = 0.6$ of player A ⁴. Then, we perform a thorough analysis of the scenarios for different settings of parameters λ and ξ and describe the influence of these parameters on the quality of the solution.

Following is the list of the scenarios and corresponding methods on how to find an optimal strategy:

- Single-agent, static \rightarrow iterate over solutions and find the one with maximal reward,
- Single-agent, dynamic \rightarrow solve MDP directly by dynamic programming,
- Multi-agent, static \rightarrow find optimum by computing Nash equilibrium,
- Multi-agent, dynamic \rightarrow learn optimum by using reinforcement learning, e.g., temporal difference algorithm.

Static strategies

We start with a static strategy, which consists of one fixed action for the entire time horizon, i.e., 100 years. A static strategy can be written down in the form of a dynamic strategy in the MDP, where at every time step the player chooses the same action.

Single-agent static Firstly, we look on a system with a single agent. Getting an optimal strategy for a single-agent static scenario is straightforward due to the discrete and small action space. We have $|A|$ possible strategies for which we compute the rewards and choose the one with the highest reward. In Table 4.2 we show the optimal strategies for different levels of λ . They are optimal in sense of maximising the total reward (payoff) defined as $r = -\left(\mathbb{E}(n_{lost}) + \lambda\mathbb{E}(n_{rem})\right)$.

⁴In case of having only two players, we obtain the share of important assets of the other player as $\xi_B = 1 - \xi_A$.

λ	0.1	0.2	0.3	0.4	0.5
strategy	3,000	5,000	5,000	9,000	9,000
total reward r	-23.3867	-27.9443	-30.0443	-31.9917	-32.74

Table 4.2: Optimal single-agent static strategies for different parameter λ , where the strategy is fixed for the entire time horizon. For increasing λ (i.e. object removal gets more costly) the optimal strategy is to remove fewer objects (i.e. greater threshold for removal).

Multi-agent static We now move to a multi-agent scenario with static strategies. This scenario can be written down as a normal-form game, the pure strategies are defined by the *threshold for removals*. For a normal-form game the solution concept is Nash equilibrium. In our analysis we assume two-players A and B. The players differ only in size defined by the share of important assets. We assume that agent i has a share ξ_i of total important assets from all important assets in the environment. The reward (payoff) for player A is defined as $r_A = \xi_A \mathbb{E}(n_{lost}) + \lambda \mathbb{E}(n_{rem}^A)$ and for player B as $r_B = (1 - \xi_A) \mathbb{E}(n_{lost}) + \lambda \mathbb{E}(n_{rem}^B)$. Note that $\xi_A = 1 - \xi_B$. The pure strategies are the *thresholds for removal* which give us the values of $\mathbb{E}(n_{lost})$ and $\mathbb{E}(n_{rem})$. We are now able to construct the payoff matrix for given parameters λ and ξ_A .

We demonstrate forming the payoff matrix for $\lambda = 0.1$ and $\xi_A = 0.6$. We show the payoff matrix for player A choosing pure strategy “*threshold for removal* = 2,000” as expected number of removed objects $\mathbb{E}(n_{rem}^A)$ and player B choosing pure strategy “*threshold for removal* = 6,000” as $\mathbb{E}(n_{rem}^B)$. We obtain the values of $\mathbb{E}(n_{rem}^{2000})$ and $\mathbb{E}(n_{rem}^{6000})$ from the simulation data and compute the value of $\mathbb{E}(n_{lost})$ by finding a threshold curve for given joint action which is a sum of removal efforts as described in Section 4.3.2. Then, we compute the payoff matrix entries r_A and r_B for the chosen pure strategies as $r_A = 0.6 \mathbb{E}(n_{lost}) + 0.1 \mathbb{E}(n_{rem}^A)$ and $r_B = 0.4 \mathbb{E}(n_{lost}) + 0.1 \mathbb{E}(n_{rem}^B)$. We state the full payoff matrix in Table 4.3, we can see how easy/difficult is to reach the respective pure Nash equilibria which we state in bold. In Table 4.4 we can see the corresponding Nash equilibria (NE), the strategy is written in format [$< player > : < action >, < probability >$]. We show all the pure equilibria (1, 2, 3) and one mixed equilibrium (4), note there exist more mixed equilibria, however we show only one. In general we are interested only in pure strategies, because in space debris removal problem it is unfeasible to play mixed strategies. For each Nash equilibrium we show the player A reward r_A , the player B reward r_B , the welfare ω (the sum of rewards) and the fairness ϕ .

	<i>3000</i>	<i>4000</i>	<i>5000</i>	<i>6000</i>	<i>8000</i>	<i>9000</i>	<i>10000</i>	<i>NR</i>
3000	-16.36 <i>-12.85</i>	-16.36 <i>-10.24</i>	-16.36 <i>-9.12</i>	-16.36 <i>-8.58</i>	-16.36 <i>-8.01</i>	-16.36 <i>-7.77</i>	-16.36 <i>-7.62</i>	-16.36 <i>-7.02</i>
4000	-13.76 <i>-12.85</i>	-13.76 <i>-10.24</i>	-16.19 <i>-10.74</i>	-16.19 <i>-10.20</i>	-16.19 <i>-9.64</i>	-16.19 <i>-9.39</i>	-16.19 <i>-9.24</i>	-16.19 -8.64
5000	-12.64 <i>-12.85</i>	-15.07 <i>-11.86</i>	-15.07 <i>-10.74</i>	-15.07 <i>-10.20</i>	-16.35 <i>-10.49</i>	-16.35 <i>-10.25</i>	-16.35 <i>-10.09</i>	-16.35 <i>-9.50</i>
6000	-12.09 <i>-12.85</i>	-14.52 <i>-11.86</i>	-14.52 <i>-10.74</i>	-15.80 <i>-11.05</i>	-15.80 <i>-10.49</i>	-15.80 <i>-10.25</i>	-15.80 -10.09	-17.34 <i>-10.52</i>
8000	-11.53 <i>-12.85</i>	-13.96 <i>-11.86</i>	-15.24 <i>-11.60</i>	-15.24 <i>-11.05</i>	-16.77 <i>-11.51</i>	-16.77 <i>-11.27</i>	-16.77 <i>-11.12</i>	-18.61 <i>-11.75</i>
9000	-11.28 <i>-12.85</i>	-13.71 <i>-11.86</i>	-14.99 <i>-11.60</i>	-14.99 <i>-11.05</i>	-16.53 <i>-11.51</i>	-18.37 <i>-12.50</i>	-18.37 <i>-12.34</i>	-18.15 <i>-11.60</i>
10000	-11.13 <i>-12.85</i>	-13.56 <i>-11.86</i>	-14.84 <i>-11.60</i>	-14.84 <i>-11.05</i>	-16.38 <i>-11.51</i>	-18.22 <i>-12.50</i>	-18.22 <i>-12.34</i>	-18.63 <i>-12.03</i>
NR	-10.54 <i>-12.85</i>	-12.97 <i>-11.86</i>	-14.25 -11.60	-15.78 <i>-12.08</i>	-17.62 <i>-12.74</i>	-17.40 <i>-12.35</i>	-18.04 <i>-12.62</i>	-20.14 <i>-13.42</i>

Table 4.3: Payoff matrix for parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$ (i.e. row player owns 60% of all assets). In normal font are the row player's payoffs, in italic are the column player's payoffs. In bold are pure Nash equilibria.

<i>NE</i>	1	2	3	4
strat	A: 4k,1 B: NR,1	A: NR,1 B: 5k,1	A: 6k,1 B: 10k,1	A: 4k,0.36; NR,0.64 B: 4k,0.61; 9k,0.39
r_A	-16.19	-14.25	-15.80	-14.13
r_B	-8.64	-11.60	-10.09	-10.89
ω	-24.83	-25.84	-25.90	-25.02
ϕ_A	0.920	1.088	0.984	1.062
ϕ_B	1.150	0.891	1.027	0.919
ϕ	0.230	0.197	0.043	0.143

Table 4.4: Optimal multi-agent static strategies, where the solution concept is Nash equilibria. We show player A's and B's rewards, welfare and fairness for parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$ (i.e. player A owns 60% of all assets). There are three pure Nash equilibria and several mixed ones (we show only one mixed NE in the last column).

λ	strategy	total reward (welfare)
0.1	3k,3k,3k,3k,3k,3k,3k,3k,3k	-23.39
0.2	5k,5k,5k,5k,4k,4k,4k,4k,5k,4k	-27.49
0.3	9k,8k,6k,5k,5k,5k,6k,5k,5k,5k	-29.69
0.4	9k,9k,10k,NR,10k,9k,9k,9k,10k,9k	-31.47
0.5	10k,NR,NR,NR,10k,10k,9k,9k,10k,9k	-32.05

Table 4.5: Optimal single-agent dynamic strategies for different parameter λ . For increasing λ (i.e., object removal gets more costly) the optimal strategy is to remove fewer objects (i.e., greater thresholds for removal) and the welfare decreases.

Dynamic strategies

We can now move to dynamic strategies as described in Section 4.3.3. The players can decide on an action every time step. Thus, the strategy is defined as a sequence of actions over the time steps. Allowing the agents to dynamically shape their strategy is more realistic than fixing the strategy through the course of the time horizon. However, dynamic strategies are severely more complex, making the whole interaction with the system and potentially with other players much more complicated. As stated before we assume time horizon 100 years with decision time steps 10 years, thus having 10 decision points, where the agent(s) have to choose an action. We describe and experiment with the single-agent and multi-agent cases in turn.

Single-agent dynamic In the single-agent case the optimal strategy is obtained by solving the underlying MDP. This is a useful property of the proposed surrogate model, where we can effectively plan the optimal strategy. For small state spaces we can iterate over the whole space and find the optimal strategy, for larger state spaces we can use dynamic programming and for even larger state spaces we can use reinforcement learning methods. We show in Table 4.5 the optimal strategies for different levels of parameter λ . The strategies are shown as a sequence of actions, e.g., the optimal strategy for $\lambda = 0.1$ is choosing the action “*threshold for removal 3,000*” in every time step. For increasing λ (i.e., object removal gets more costly) the optimal strategy is to remove fewer objects (i.e., greater thresholds for removal), resulting in a decrease in welfare. We compare Table 4.5 with Table 4.2 and can see that the dynamic strategies are better than (or at least as good as) the static strategies in terms of total reward (welfare), this result is intuitive because the player has more flexibility in the dynamic case.

λ	strategy	total reward (welfare)	Δ
0.1	3k,3k,3k,3k,3k,3k,3k,3k,3k	-23.39	0 %
0.2	4k,4k ,5k,5k,4k,4k,4k,4k,5k,4k	-27.64	0.5 %
0.3	6k,8k ,6k,5k,5k,5k,6k,5k,5k,5k	-29.69	0 %
0.4	10k,NR,NR ,NR,10k, 10k ,9k,9k,10k,9k	-31.52	0.2 %
0.5	10k,NR,NR,NR,10k,10k,9k,9k,10k,9k	-32.05	0 %

Table 4.6: Learned single-agent dynamic strategies for different λ . Differences to the optimal strategies from Table 4.5 are shown in bold and differences in rewards are stated in the last column. We can successfully validate the learning process due to high similarity (very low differences) to the optimal strategies.

In the multi-agent scenario, or in the case of larger state space it might be unfeasible or computationally demanding to explicitly solve the MDP and compute the optimal strategy. Thus, we also discuss learning the optimal strategy. We use a standard reinforcement learning method Q-learning as described in Section 2.2.3. In Table 4.6 we show the single-agent dynamic strategies which we learned using Q-learning and compare them with the optimal ones from Table 4.5 to validate the learning method. In bold we can see the differences to the optimal strategies and in the last column we state the difference in total reward between the learned and the optimal strategies. We can see that especially at the beginning the learned strategies might differ; this is caused by rather similar threshold curves behaviour at the beginning of the time horizon (see Figure 4.1). We can conclude that the learning method is successful and we will use it for further analysis in the multi-agent scenario.

Multi-agent dynamic In the space debris removal problem we can consider several space actors interacting with each other and deciding on a removal strategy. Therefore, from a single-agent, we move to a multi-agent dynamic scenario. We now face a very difficult problem of finding the optimal strategies due to the *moving target problem* [Tuyts and Weiss, 2012] and potentially very large state space. Thus, we focus on learning the optimal strategies. We first show learning a strategy against a fixed opponent (opponent playing a static strategy) and then learning a strategy against a learning strategy. In our analysis we assume two players, which we denote player A (primarily the learning agent) and player B (opponent). The players differ only by their share of important assets ξ , which represents the space actor size. We make here a model design assumption of identical space

program of different space actors differing only in size, i.e., homogeneous spacecraft types, spacecraft sizes, used orbits, etc.

Multi-agent dynamic: Against fixed opponent We assume the opponent (player B) has a fixed strategy, which is one of the possible thresholds; this is a static strategy as described above. We show in the Table 4.7 learned strategies against different fixed strategies. In the first column we state the opponent (player B) fixed strategy, e.g., 3k means the player B will choose in every time step the threshold 3,000. We compare two types of learned strategies for the player A: (i) altruistic strategy, which maximise the environment welfare and (ii) selfish strategy, which maximises the player A’s reward. In the Table 4.7 we also state the players’ rewards r_i , welfare ω , price of anarchy PoA between altruistic and selfish behaviour of player A and fairness ϕ . We can see that the price of anarchy is similar for most of the fixed strategies. This means that once the opponent fixes his strategy the environment welfare can be improved by approximately 5% – 6% whether we play selfishly or altruistically. Finally, we show the fairness ϕ as described in Section 4.3.3. We can observe that the selfish behaviour is fairer compared to the altruistic one, which is expected. We can also see that some of the fixed strategies give very bad environment welfare, e.g., “fixed 1k” gives more than double the loss.

Multi-agent dynamic: Learning against learning strategy We discussed before that the multi-agent scenario is too complex to compute an optimal strategy. Therefore, we now investigate the dynamics of two players learning each other’s strategy. We assume both players learning the strategy by using the standard Q-learning algorithm. We assume discount factor $\gamma = 1$, i.e., no discount, the learning rate $\alpha = 0.01$, and the exploration parameter $\epsilon = 0.1$. We discretise the state space as described in Section 4.3.2 to debris levels with step size 1,000 and a time step of 10 years. We learned all the strategies over 1 million episodes. Both players can fully observe the state, and the Q-values are independent on the other player action, thus this learning can be seen as independent Q-learning, which is a common method in multi-agent reinforcement learning [Bloembergen et al., 2015].

In Table 4.8 we show several learned strategies for a single setting of the parameters $\lambda = 0.1$ and $\xi_A = 0.6$. We show four different outcomes of the same setting. We can see the strategies for player A and B, their rewards r_i , the welfare ω and the individual and overall fairness ϕ . We can see that we can attain highly effective solutions using Q-learning for both players. One can note that these strategies have lower welfare ω than the worst pure

fixed B	type	strategy	r_A	r_B	ω	PoA	ϕ
1k	altr	10k,NR,NR,NR,NR,NR,NR,NR,NR,NR	-3.70	-43.27	-46.97	-	7.19
1k	self	10k,NR,NR,NR,NR,NR,NR,NR,NR,NR	-3.70	-43.27	-46.97	1	7.19
3k	altr	10k,NR,NR,NR,NR,NR,NR,NR,NR,NR	-10.54	-12.85	-23.39	-	0.60
3k	self	10k,NR,NR,NR,NR,NR,NR,NR,NR,NR	-10.54	-12.85	-23.39	1	0.60
5k	altr	3k,3k,4k,4k,3k,4k,3k,3k,3k,3k	-14.64	-8.75	-23.39	-	0.111
5k	self	10k,NR,10k,9k,8k,8k,8k,6k,6k,6k	-14.07	-10.58	-24.65	1.054	0.115
6k	altr	3k,3k,3k,3k,3k,3k,3k,3k,3k,3k	-15.14	-8.25	-23.39	-	0.207
6k	self	5k,5k,5k,6k,5k,6k,6k,5k,6k,6k	-14.65	-10.11	-24.76	1.059	0.034
8k	altr	3k,3k,3k,3k,3k,3k,3k,3k,3k,3k	-15.52	-7.87	-23.39	-	0.285
8k	self	5k,5k,6k,6k,5k,5k,5k,4k,5k,4k	-15.07	-9.61	-24.68	1.055	0.045
10k	altr	3k,3k,3k,3k,3k,3k,3k,3k,3k,3k	-15.83	-7.56	-23.39	-	0.351
10k	self	5k,5k,6k,5k,4k,4k,4k,4k,5k,4k	-15.44	-9.24	-24.68	1.055	0.109
NR	altr	3k,3k,3k,3k,3k,3k,3k,3k,3k,3k	-16.36	-7.02	-23.39	-	0.475
NR	self	5k,5k,5k,5k,4k,4k,4k,4k,5k,4k	-15.93	-8.75	-24.68	1.055	0.199

Table 4.7: Optimal multi-agent dynamic strategies against fixed opponent for parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. We show optimal altruistic (altr) and selfish (self) strategies. In the first column we show the opponent (player B) fixed strategy. We state the rewards, welfare, fairness and price of anarchy between different solutions. We can see that fixed strategies can lead to very sub-optimal solutions.

player	strategy	r_i	ω	ϕ_i	ϕ
A	6k,6k,5k,6k,5k,4k,4k,4k,5k,4k	-15.51	-24.75	0.957	0.113
B	9k,8k,6k,8k,8k,9k,10k,NR,NR,NR	-9.25		1.070	
A	6k,5k,4k,4k,4k,4k,4k,4k,4k,4k	-16.00	-24.76	0.929	0.202
B	9k,10k,9k,10k,NR,NR,NR,NR,NR,NR	-8.76		1.131	
A	6k,8k,6k,5k,4k,4k,4k,4k,5k,4k	-15.84	-24.83	0.941	0.164
B	10k,9k,9k,10k,NR,NR,NR,NR,NR,10k	-8.99		1.105	
A	6k,8k,8k,6k,5k,4k,4k,4k,5k,4k	-15.62	-24.91	0.957	0.116
B	9k,10k,9k,9k,8k,9k,10k,NR,NR,NR	-9.29		1.073	

Table 4.8: Learned multi-agent dynamic strategies using Q-learning against Q-learning opponent with parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. We show four different outcomes of the same setting. We can attain highly effective solutions using Q-learning for both players.

Nash equilibrium welfare in static scenario (Table 4.4). We can compare these strategies in terms of fairness ϕ or welfare ω , where these two metrics are not necessarily dependent on each other. A better welfare does not mean fairer division of removal efforts.

So far we have shown the learning process for a fixed size of the players represented by the parameter of assets share ξ . We now investigate how different levels of ξ influence the solution and its quality. We show such an analysis in Table 4.9. We experiment with 9 different divisions of shares of important assets between the two players, in the first column we show the shares ξ for each of the players. We can see that the less a player owns the less he wants to remove and vice versa, which is expected. One can see that for the cases when a player owns only a small proportion of the assets he prefers not to remove anything, e.g., $\xi_i = 0.1$. A very important outcome from this table is the evolution of the environment welfare, one can observe that for disproportional players we get higher welfare than for proportional players (compare welfare of $\xi_A = 0.1$ to welfare of $\xi_A = \xi_B = 0.5$, -23.39 and -25.07 respectively). Another important outcome is how the size of the players influence the fairness. Looking at the Table 4.9 we can note that the more similarly sized the players are the fairer strategy they can learn. In the case of very disproportional players, they learn very unfair strategy, e.g., $\xi_i = 0.1$ or $\xi_i = 0.2$. For equally sized players, the learned strategy is the most fair. Thus, high disproportion in the players' size attains high welfare but trades off for fairness, where the small sized player removes barely anything.

ξ_i	player	strategy	r_i	ω	ϕ_i	ϕ
0.1	A	10k,10k,NR,NR,NR,NR,NR,NR,NR,NR	-1.76	-23.39	1.329	0.356
0.9	B	3k,3k,3k,3k,3k,3k,3k,3k,3k,3k	-21.63		0.973	
0.2	A	10k,10k,NR,NR,NR,NR,NR,NR,NR,NR	-3.57	-23.41	1.311	0.368
0.8	B	3k,3k,3k,4k,3k,3k,3k,3k,3k,3k	-19.85		0.943	
0.3	A	10k,NR,NR,NR,NR,NR,NR,NR,NR,NR	-6.42	-24.47	1.143	0.194
0.7	B	5k,5k,4k,5k,4k,4k,4k,4k,4k,4k	-18.05		0.949	
0.4	A	NR,NR,10k,NR,NR,NR,NR,NR,NR,NR	-8.78	-24.68	1.124	0.193
0.6	B	5k,6k,5k,5k,4k,4k,4k,4k,5k,4k	-15.90		0.931	
0.5	A	6k,8k,8k,6k,6k,5k,6k,6k,8k,6k	-12.46	-25.07	1.006	0.012
0.5	B	8k,8k,8k,6k,5k,6k,6k,5k,6k,5k	-12.61		0.994	
0.6	A	6k,8k,8k,6k,5k,4k,4k,4k,5k,4k	-15.62	-24.91	0.957	0.116
0.4	B	9k,10k,9k,9k,8k,9k,10k,NR,NR,NR	-9.29		1.073	
0.7	A	5k,4k,4k,5k,4k,4k,4k,4k,4k,4k	-18.12	-24.55	0.948	0.197
0.3	B	NR,10k,NR,NR,NR,NR,NR,NR,NR,NR	-6.43		1.145	
0.8	A	3k,2k,3k,3k,3k,3k,3k,3k,3k,3k	-20.08	-23.58	0.939	0.405
0.2	B	NR,10k,NR,NR,NR,NR,NR,NR,NR,NR	-3.51		1.344	
0.9	A	3k,3k,3k,3k,3k,3k,3k,3k,3k,3k	-21.63	-23.39	0.973	0.356
0.1	B	NR,NR,NR,NR,NR,NR,NR,NR,NR,NR	-1.76		1.329	

Table 4.9: Learned multi-agent dynamic strategies using Q-learning against Q-learning opponent with parameter $\lambda = 0.1$ and different levels of assets share ξ_i . Starting from highly disproportional players in the top row to equally sized players in the middle row. High disproportion in the players' size attains high welfare but trades off for fairness, where the small sized player removes barely anything.

code	agents	type	obtained	ω_{max}	ω_{min}	ϕ_{min}	ϕ_{max}
SO1	1	static	optimal	-23.39	-23.39	-	-
DO1	1	dynamic	optimal	-23.39	-23.39	-	-
DL1	1	dynamic	learned	-23.39	-23.39	-	-
SO2	2	static	optimal (NE)	-24.83	-25.90	0.043	0.230
FO2	2	dyn/fixed	optimal	-23.39	-46.97	0.034	7.19
DL2	2	dynamic	learned	-24.75	-24.91	0.113	0.202

Table 4.10: Comparison of different scenarios in terms of welfare ω and fairness ϕ for $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. We show combinations of single-agent, multi-agent, static and dynamic approaches which were obtained either by learning or by exact computation. In case there were multiple solutions for given scenario we present maximal and minimal values.

Analysis and comparison of different strategy concepts

We have presented several scenarios using the surrogate model and have described methods how to find efficient strategies for them. We now compare the different solutions of those scenarios in terms of quality and efficiency. In Table 4.10 we show the different scenarios and their values of welfare ω and fairness ϕ . We state single-agent, multi-agent, static and dynamic scenarios and the methods to find respective effective strategies. For every scenario we state a code, which we use in further analysis. Note that the optimal strategies were obtained by exhaustive search (brute force) and the learned strategies by Q-learning. All the shown combinations of scenarios were discussed in turn in the previous sections. We compare here only the scenarios for parameters setting of $\lambda = 0.1$, $\xi_A = 0.6$ and time step 10 years in a 100 year horizon. For some of the scenarios we obtained several solutions, thus we state maximal and minimal values of welfare and fairness. We can see that playing against a fixed opponent can cause a high unfairness, which is expected due to the non-optimal fixation of removal effort.

We now focus on comparing these different scenarios and on how efficient they are. For such comparison we use the concept of price of anarchy PoA as described in Section 4.3.3. We assume two types of PoA , the first type PoA_m compares the single-agent scenario with the multi-agent one, i.e., the cost for having self interested (competing) players instead of centralised (single-agent) strategy and PoA_d which compares the static scenario with the dynamic one. PoA_d can be thought of as the advantage we get by playing dynamically, i.e., being able to change the strategy in every time step. In Table 4.11 we compare all

code	SO1	DO1	DL1	SO2	FO2	DL2
SO1	1	1	1	<i>1.107</i>	2.008	1.058
DO1	1	1	1	1.107	2.008	<i>1.058</i>
DL1	1	1	1	1.107	2.008	1.058
SO2	<i>1.107</i>	1.107	1.107	1	1.814	1.047
FO2	2.008	2.008	2.008	1.814	1	1.898
DL2	1.058	<i>1.058</i>	1.058	1.047	1.898	1

Table 4.11: Comparison of different scenarios in terms of price of anarchy PoA for $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. The codes of the scenarios are stated in Table 4.10. In **bold** we show PoA_d (static vs. dynamic) and in *italic* we show PoA_m (single-agent vs. multi-agent). Note that for example $PoA = 1.107$ means 10.7% inefficiency.

the scenarios (code names from Table 4.10) in terms of price of anarchy for fixed $\lambda = 0.1$ and $\xi_A = 0.6$. Of interest are the values in bold and in italic, which show price of anarchy PoA_d between the static and dynamic scenario and price of anarchy PoA_m between the single- and multi-agent scenario, respectively. One can note that the cost of using a static strategy over a dynamic one (SO2 vs. DL2) is 4.7% for the multi-agent case and 0% for the single-agent case (SO1 vs. DO1). The cost of multi-agent scenario to single-agent scenario is 5.8% for the dynamic case (DO1 vs. DO2) and 10.7% for the static case (SO1 vs. SO2), which is caused by the selfish behaviour of the players. Also note the high values of PoA of the multi-agent fixed scenarios (FO2), meaning that a fixed strategy can cause a highly inefficient outcome in the terms of the environmental welfare.

We have shown the methodology of comparison of different scenarios for fixed parameters of λ and ξ . In the next section we investigate the quality of the scenarios for varying levels of those parameters.

Varying levels of ξ and λ

We investigate the different scenarios and corresponding optimal solutions for different settings of the two main studied parameters: (i) ratio λ between the cost of removal C_R and the cost of losing an important asset C_L and (ii) share of important assets ξ . We do the analysis for $\lambda \in [0.1, 0.2, 0.3, 0.4, 0.5]$ and $\xi \in [0.1, 0.2, 0.3, 0.4, 0.5]$. As stated before the players differ only in the size expressed by the parameter ξ , thus the results for $\xi_A = 0.4$ and $\xi_B = 0.6$ in the two player case are the same as $\xi_A = 0.6$ and $\xi_B = 0.4$. Obviously, this holds for any setting of ξ . We run the experiments for the 4 main scenarios; single-agent static,

λ	0.1	0.2	0.3	0.4	0.5
static	-23.39	-27.94	-30.04	-31.99	-32.74
dynamic	-23.39	-27.49	-29.69	-31.47	-32.05
PoA_d	1	1.016	1.012	1.017	1.022

Table 4.12: Comparing static and dynamic single-agent scenarios in terms of welfare ω and price of anarchy PoA_d for varying parameter λ . Note that for increasing λ (i.e., cost of removal becomes more expensive) the welfare decreases and the difference between a static and a dynamic scenario increases.

single-agent dynamic, multi-agent static and multi-agent dynamic. The methods to obtain optimal strategies for these scenarios are discussed in the previous sections. The comparison metric is the price of anarchy PoA . We distinguish between comparing single-agent with multi-agent scenarios using PoA_m and static with dynamic scenario using PoA_d .

Firstly, we compare single-agent scenarios, in Table 4.12 we show the welfare ω and PoA_d for static and dynamic scenarios for different levels of λ . We can observe that by using the dynamic strategy we can improve the environment welfare by up to 2.2% (in the case of $\lambda = 0.5$). Thus, for increasing λ (i.e., cost of removal becomes more expensive) the welfare decreases and the difference between a static and a dynamic scenario increases. Moving to the multi-agent case in Table 4.13 we first show the optimal solutions to static scenario obtained by computing the Nash equilibria. We mentioned before that we are only interested in the pure Nash equilibria, thus in the case of multiple pure equilibria for given parameters ξ_A and λ we show only the minimal and maximal values of those in the Table 4.13. One can see that some of the values are repetitive, this is caused by limited flexibility of the static solutions and potentially by the available actions. For instance, the high values of parameter λ mean that it is very expensive to remove objects compared to losing assets, meaning that the players prefer to remove as few as possible, e.g., no-removal strategy, hence some of the constant welfares in the table. One can see that the values of welfare ω decrease with increasing values of λ and with more equally sized players expressed by the share parameter ξ . Although, this conclusion is achieved only experimentally, it does strongly suggest such a trend. The similarly sized players cause inefficiency of the environment welfare due to being selfish. As discussed before, obtaining the optimal strategies for the single-agent scenario is not computationally as demanding as for the multi-agent scenario, where we might not be able to compute the optimal strategy but need to learn it. In Table 4.13 we state the resulting welfares ω for multi-agent dynamic

	$\lambda \backslash \xi_A$	0.1	0.2	0.3	0.4	0.5
Nash Eq. static	0.1	-23.39	-27.94	-30.04	-31.99	-32.74
	0.2	-23.39	-27.94	-30.04	-31.99	-33.56
	0.3	-23.39/-25.90	-27.94	-31.24	-31.99	-33.56
	0.4	-24.83/-25.90	-27.94	-31.24	-33.56	-33.56
	0.5	-25.84/-25.90	-28.05/-30.50	-31.24	-33.56	-33.56
Q-learned dynamic	0.1	-23.39	-27.65	-29.72	-31.53	-32.07
	0.2	-23.41	-27.71	-29.74	-31.54	-32.06
	0.3	-24.91	-27.86	-30.99	-31.53	-32.86
	0.4	-24.77	-27.86	-30.99	-31.65	-32.34
	0.5	-25.47	-28.22	-31.17	-32.02	-32.86

Table 4.13: Comparison of static and dynamic multi-agent scenarios in terms of welfare ω for different levels of λ and share parameter ξ . The static scenario is obtained by computing Nash equilibria and the dynamic scenario is learned using Q-learning. In case of multiple solutions we state maximal and minimal values (multiple NE). Note that with increasing parameter λ (object removal becomes more expensive) and increasing ξ (the players become more equally sized) the welfare decreases. One can see the improvement in welfare of dynamic strategies compared to the static ones.

scenario and varying levels of the studied parameters. We can again see the same trend; with increasing λ and ξ the welfare worsens.

We now have the welfare ω values for all the scenarios and all the settings of the studied parameters. We are interested in comparing them in terms of price of anarchy PoA , which expresses the inefficiency between different scenarios. We start with PoA_m comparing single-agent static (Table 4.12) with multi-agent static (Table 4.13) scenarios in Table 4.14. We can see that the inefficiency induced by having multiple players is ranging from 0% to 10.7%. One can observe that the inefficiency grows with more equally sized players, which is to be expected. From static scenarios we move to comparing dynamic scenarios, in Table 4.14 we show the analysis of PoA_m , comparing single-agent dynamic with multi-agent dynamic scenario. We obtain inefficiencies ranging from 0% to 8.9%. One can again see that PoA_m grows with more equally sized players, where we get the highest values for the same sized players, i.e., $\xi_i = 0.5$. This is the cost to pay for competing selfish agents compared to having a centralised solution, i.e., a single entity deciding on removal effort.

Finally, we look at PoA_d between multi-agent static (NE) and multi-agent dynamic (Q-learned) in Table 4.15. As expected the dynamic solutions are better than the static ones,

$\xi_A \backslash \lambda$	0.1	0.2	0.3	0.4	0.5
0.1	1.000	1.000	1.000	1.000	1.000
0.2	1.000	1.000	1.000	1.000	1.025
0.3	1.107	1.000	1.040	1.000	1.025
0.4	1.107	1.000	1.040	1.049	1.025
0.5	1.107	1.092	1.040	1.049	1.025

Table 4.14: Comparison of single-agent and multi-agent static scenarios in terms of price of anarchy PoA_m for varying levels of share parameter ξ and parameter λ . We can observe the increasing inefficiency of solutions for increasing ξ (the players become more equally sized).

except for the setting $\lambda = 0.1$ and $\xi_A = 0.2$, which is caused by learning only a sub-optimal strategy. We can expect that with a higher number of episodes we would obtain better dynamic strategies than in the static case even for this setting of the parameters. We can observe that the solutions of static vs. dynamic scenarios differ from 0% to 8.1%. Thus, the inefficiency in the multi-agent scenario induced by being limited to a static strategy compared to a dynamic strategy can be up to 8.1%.

4.5 Discussion

In this chapter we have presented a new approach to multi-agent modelling and learning of a complex environment of the space debris removal problem. We have introduced several models of agent interaction, where space agencies decide on removal strategies, determining the future evolution of the space debris environment.

Firstly, we implemented a realistic high-fidelity simulation model of earth orbit environment based on the PyKEP scientific library. We developed a collision model, a breakup model and two different launch models; (i) a simple launch model, which is consistent with related work and assume a simple repeating of past launch sequences and (ii) a complex launch model, which considers different spacecraft classes, follows the proposed mitigation guidelines [Inter-Agency Space Debris Coordination Committee, 2007] and enables to model various future scenarios of technological development. We fully describe this simulator in Appendix A. Our experiments with the simulator confirmed the commonly predicted exponential growth of space debris in near earth orbits. This is an important motivation to come up with mitigation strategies such as active object removal.

	$\lambda \backslash \xi_A$	0.1	0.2	0.3	0.4	0.5
PoA_m	single-agent dynamic vs. multi-agent dynamic					
	0.1	1.000	1.006	1.001	1.002	1.001
	0.2	1.001	1.008	1.002	1.002	1.000
	0.3	1.065	1.013	1.044	1.002	1.025
	0.4	1.059	1.013	1.044	1.006	1.009
	0.5	1.089	1.027	1.050	1.018	1.025
PoA_d	multi-agent static vs. multi-agent dynamic					
	0.1	1.000	1.010	1.011	1.015	1.021
	0.2	0.999	1.008	1.010	1.014	1.047
	0.3	1.040	1.003	1.008	1.015	1.021
	0.4	1.046	1.003	1.008	1.060	1.038
	0.5	1.017	1.081	1.002	1.048	1.021

Table 4.15: Comparison of single-agent dynamic vs. multi-agent dynamic and multi-agent static vs. multi-agent dynamic scenarios in terms of price of anarchy (PoA_m and PoA_d) for varying levels of share parameter ξ and parameter λ . Note that in the comparison of the single-agent dynamic vs. multi-agent dynamic scenarios for increasing parameter ξ (more equally sized players) the inefficiency increases.

Using statistics from extensive Monte Carlo roll-outs from the developed full simulator with complex launch model we proposed a computationally efficient surrogate model that accurately captures the dynamics of the space debris environment for various debris removal strategies. Unlike other surrogate models in the literature [Lewis et al., 2009], we derived our surrogate model by curve-fitting the full simulation results including various launch scenarios and accurately simulated orbital motion. This ensures that our surrogate model faithfully represents our full simulation, without the potential bias introduced by a specific choice of surrogate model parameters, such as a fixed insertion rate of debris. We have shown various ways in which this surrogate model can be used to study the effect of different strategies. Apart from considering a static one-shot interaction in the form of a normal-form game, we also investigated dynamic strategies, and studied the resulting high dimensional complex strategic interaction. This is a novel contribution in the area of debris removal, where previous studies on the cost of removal considered either the effect of cooperatively removing individual objects or using simple, fixed strategies for each actor [Liou and Johnson, 2009; Liou et al., 2010, 2013]. In addition, we have formulated a stochastic game based on the surrogate model, which we used to study multi-party decision

making. As an example, we have shown how machine learning techniques (here, Q-learning) can be used to *learn* an optimal debris removal strategy that outperforms fixed strategies. We have compared and evaluated both a single-agent and a multi-agent approach to the problem of space debris removal. By computing the price of anarchy we analysed the cost of decentralised (individually rational) decision making as compared to a centrally optimised strategy. Our results showed that such cost can be up to 10.7% in the static case and up to 8.9% in the dynamic case depending on the parameters of ratio λ between cost of removal and cost of losing an important asset and share ξ of important assets defining the size of the players. We can see that the cost of decentralised solution is quite significant, considering the enormous level of resources needed for the space debris removal. Thus, the space actors should aim to minimise the number of competing agents in the environment by for example forming coalitions.

Furthermore, we investigated the difference between static strategies and dynamic strategies. Static strategies have the advantage of simplicity of the decision making, but are less effective than their dynamic counterparts. We compared both in terms of price of anarchy. In the single-agent case, the cost of using a static strategy is up to 2.2%, and for the multi-agent case the cost is up to 8.1% depending on the setting of parameter λ . Comparing single-agent vs. multi-agent scenarios and static vs. dynamic scenarios we showed that the parameter ξ – the share of important assets, representing the size of the players – has a big impact on quality of the solution. The more similarly sized the players are the less efficient solutions we obtain, i.e., equally sized players produce the worst solutions. On the other hand, highly disproportional players arrive at more efficient solutions and the values of price of anarchy PoA for single-agent vs. multi-agent and static vs. dynamic scenarios are equal or very close to 1, meaning there is no or low inefficiency. We were also interested in fairness of the players' strategies depending on their size. The idea of fairness was driven by the assumption that the level of the removal effort should be proportional to the size of the player. In our analysis we defined the concept of fairness and described how the size of the players (given by their number of assets) influences the final outcome in terms of global welfare and fairness. We found out that the more equally sized the players are the fairer strategy can be learned at the cost of reduced global welfare. On the other hand, the more disproportional the players are the better global welfare they can attain, at the cost of a more unfair distribution of effort. This realisation is in line with the increasing price of anarchy for more selfishly acting players.

This result in particular might serve to inform policy and decision making processes.

A coordinated, global approach towards space debris removal, effectively reducing to one single actor, may be more effective in maximising the effect on the space environment than the current, distributed approach of various actors acting independently. Such a global entity for space debris removal could be set up through international agreements with proportional contributions by different actors, thus maintaining fairness while achieving a maximum of impact.

With respect to the problem statement of this thesis presented in Section 1.2, we have proposed a new methodology to model complex environments. While such methodology is often highly domain specific, general guidelines can be derived. Modelling of complex environments such as the space debris removal problem requires a wide scale of assumptions and modelling choices to make. We have proposed one potentially effective approach to multi-agent modelling and learning, starting from simple models (one-shot interaction), building on it to more complex models (dynamic multi-stage interaction). We investigated important metrics to evaluate like social welfare or fairness for different types of solutions.

5

Robust Learning in Critical Systems with Risky States

This chapter is based on the following publications:

- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2018a). Learning robust policies when losing control. *Adaptive and Learning Agents workshop at AAMAS*
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2018b). Towards learning to best respond when losing control. *European Workshop on Reinforcement Learning (EWRL)*, pages 1–11
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2019). Robust temporal difference learning for critical domains. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 350–358

5.1 Need for Robustness in Systems with Risky States

In this chapter we discuss how a robust learning process can be designed in a complex multi-agent environment. We focus on critical domains with risky states. We consider a distributed control problem, for which each controller executes an individual policy on separate hardware. Any of such controllers may be attacked, potentially affecting or changing the local policy. These attacks or failures can be rare but can have a profound impact on the whole system if the system is not prepared for them. The main question is then, how to obtain effective and robust policies in such systems. The reasons for a controller executing a different policy than desired include: an attack by an adversary (e.g., hacker), a natural disaster (e.g., earthquake) or a mechanical defect (e.g., node malfunction).

We present a new approach for learning policies in such systems that are safe and robust against a chosen scenario of potential attacks or failures. We accomplish this by introducing a new Q-function operator, which we call the κ operator, that encodes robustness into the bootstrapping update of traditional temporal difference (TD) learning methods presented in Section 2.2.3. In particular, we design the operator to encode the possibility of significant rare events (SREs) without requiring the learning agent to observe such events in training. Although the κ operator is model-based with respect to these SREs, it can be combined with any TD method and can thus still be model-free with respect to the environment dynamics. We prove convergence of our methods to the optimal robust Q-function with respect to the model using the theory of Generalised Markov Decision Processes [Szepesvari and Littman, 1997]. In addition we prove convergence to the optimal Q-function of the original MDP given that the probability of SREs vanishes. Our empirical evaluations demonstrate superior performance of κ -based TD methods both in the early learning phase as well as in the final converged stage. In addition we show robustness of the proposed method to small model errors, as well as its applicability to multi-agent joint-action learning.

Therefore, we present a novel model-free approach to learn safe and robust policies in systems that are prone to be attacked, or in which some parts of the system may fail. The proposed family of algorithms opens new ways to effective behaviour in such systems, and can help mitigate some of the security threats using a priori information about the potential nature of the attack or failure. The presented learning process can thus adapt to a dynamic opponent and can use the information it gathers during interaction with the environment. In complex systems such learning can be combined with powerful function approximation techniques such as deep neural networks; we leave this extension for future work. This

chapter is structured as follows: The next part situates our approach with respect to the related work, followed by Section 5.2 which introduces the new TD operator κ , for which we subsequently prove convergence in single- and multi-agent case in Section 5.3. Section 5.4 provides empirical results further demonstrating the quality of our methods. This chapter uses several concepts from reinforcement learning and multi-agent learning; for introduction of those concepts see Section 2.2 and Section 2.3, respectively.

Related Work The aim to find robust policies is relevant to multiple research areas, including security games, robust control/learning, safe reinforcement learning and multi-agent reinforcement learning.

The domain of **security games** has expanded in recent years with many real-world applications in critical domains, examples include the ARMOR system for airport security [Pita et al., 2008] or the PROTECT system for scheduling Coast Guard [Shieh et al., 2012], where the main approach has been computing exact solutions and deriving strong theoretical guarantees, mostly using equilibria concepts such as Nash and Stackelberg equilibria [Korzhyk et al., 2011], with an extension to the Stackelberg multi-agent setting [Lou et al., 2017]. This has been very important to theoretically underpin the field, however it often seems difficult to deploy exact theoretical solution methods in real world settings due to strict model assumptions or severe simplifications [Tambe and An, 2012]. On the one hand, our work adopts the information asymmetry assumption often used in Stackelberg security games [Korzhyk et al., 2011], providing the model of attack types for the *leader*, and allowing leader-strategy-informed best response strategies by attackers. On the other, we base our approach on *reinforcement learning from interactions* with the environment, thus we do not need to know the system model. There has been substantially less work on reinforcement learning for security games than on game-theoretic approaches, with few exceptions, for example Ruan et al. [2005] who use reinforcement learning in the context of patrolling problems. Security games often assume frequent adversarial attack, whereas our work focuses on occasional loss of control over the system, which can represent e.g., failures or adversarial attacks.

Similarly to security games, control theory starts with a model of the system to be controlled (the *plant*), and for the purpose of **robust control** assumes a set of possible plants as an explicit model of uncertainty, seeking to design a policy that stabilises all these plants [Zhou and Doyle, 1998]. A slightly weaker assumption is made in related work that assumes control over the number of observations for *significant rare events* (SREs),

performing updates by sampling [Ciosek and Whiteson, 2017]. It introduces a policy gradient variant to improve learning in presence of SREs, using a *proposal distribution* that controls data from which it learns and *importance sampling* to adjust updates. While our assumption of multiple controllers could be linked to distinct plant models, we here seek to maximize in expectation by assuming probabilities of alternate controllers taking over, reducing the model back to one system. In contrast, our work assumes that the model of the system is not known a priori, and a policy needs to be learned by interacting with it, as in robust learning.

While early work on **robust reinforcement learning** focused on learning within parameterised acceptable policies [Singh et al., 1994], later work transferred the objective of maximising tolerable disturbances from control theory to reinforcement learning [Morimoto and Doya, 2005]. Our work is similar to the therein defined *Actor-disturber-critic*, but we replace its model of minimax simultaneous actions with stochastic transitions between multiple controllers (one being in control at any time) with arbitrary objectives for each controller. We thus cover not only minimising adversaries but also random failures or any other policy encoding other adversaries’ agendas.

In relation to the taxonomy of **safe reinforcement learning** of Garcia and Fernández [2015] our method falls in between *Worst-Case Criterion under Parameter Uncertainty* and *Risk-Sensitive Reinforcement Learning Based on the Weighted Sum of Return and Risk*, depending on the chosen alternate controller objectives. Our $Q(\kappa)$ method is comparable to the β -pessimistic Q-learning method of Gaskett [2003], however, we propose a more general κ operator of which $Q(\kappa)$ is only an example. Finally, our approach has commonalities with the **multi-agent reinforcement learning** algorithm Minimax-Q [Littman, 1994] for zero-sum games, which assumes minimisation over the opponent action space. However, in contrast, we define an attack to minimise over our own action space, and thus learn (but not enact) simultaneously our optimal policy and the (rare) attacks it is susceptible to. We further cover not only minimising adversaries but also random failures or any other policy encoding other adversaries’ agendas (see Section 5.2.1). While *on-policy* learning algorithms have been shown to perform better than classical Q-learning in a perturbed environment [Singh et al., 2000], and can thus in some sense be considered safer (against mistakes or exploration), our method combines with both on- and off-policy learning, and provides robustness against a chosen target.

5.2 The Robust Temporal Difference Operator κ

In this section we use the framework of temporal difference learning, described in Section 2.2.3, Markov decision processes, described in Section 2.2.1 and stochastic games, described in Section 2.3.1.

We now present our robust TD operator κ . Before we formally define the operator, we give an intuitive example. Suppose a Q-learning agent needs to learn a safe policy against a potential malicious adversary who could, with some probability \varkappa , take over control in the next state.¹ The value of the next state s_{t+1} , thus, depends on who is in control²: if the agent is in control, he can choose an optimal action that maximises expected return; or if the adversary is in control he might, in the worst case, aim to minimise the expected return. This can be captured by the following modified TD error

$$\delta_t = r_{t+1} + \gamma \left((1 - \varkappa) \max_a Q(s_{t+1}, a) + \varkappa \min_a Q(s_{t+1}, a) \right) - Q(s_t, a_t),$$

where we assume that the agent has knowledge of (or can estimate) the probability \varkappa .

In the following we first present a formal, general model of the operator κ , by modifying the target in the classical Bellman style value function. We then present practical implementations of $\text{TD}(\kappa)$ methods that use this operator for both single- and multi-agent settings, based on the classical on- and off-policy TD learning algorithms (Expected) SARSA and Q-learning.

5.2.1 Formal Model

We consider a set of m possible *control policies* $C = \{\sigma_1, \dots, \sigma_m\}$. A control policy prescribes which action is executed in a given time step and a given state, for example, action which minimises the Q-value function (adversarial control policy). At each time step, one of these policies is in control (and thus decides on the next action) with some probability $p(\sigma_i|s)$ that may depend on the state s . The set C and probability function $p(\cdot)$ are assumed to be (approximately) known by the agent. In our new TD methods,

¹We use the symbol κ to denote the proposed TD operator and the symbol \varkappa for the parameter of probability of attack, both are different versions of the letter “kappa”.

²Note that while a token of control could be included in the state (doubling its size), our approach rather directly applies model-based bootstrap updates. This makes it explicit that the robustness target is a chosen parameter of the operator, and makes it possible to learn robust strategies before observing SREs, or when learning does not occur during SREs due to compromise.

the value of the next state s' then becomes a function of both, the state and the function $p(\cdot)$, which we capture in our proposed operator κ , as $V^\kappa(s')$. Note that the set C includes the focal policy π that we seek to optimise in face of (possibly adversarial) alternative controllers. Such external control policies can represent for example a malicious attacker, aiming to minimise the expected return, or any arbitrary dynamics, such as random failures, represented by, e.g., a uniformly random policy. Based on a prior assumption about the nature of σ we want to optimise the focal policy π without necessarily observing actual attacks or failures. This means learning our robust policy π right from the start.

We define σ in terms of our own Q-value function, for example an attacker that is minimising our expected return. Thus we need to learn only one Q-value function Q^π . This is similar to the standard assumption in Stackelberg games that the attacker is able to fully observe our past actions and thus can enact the informed best response. We define the Q-value function update for our policy π based on standard Bellman equation and given the operator κ as

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha \left[\underbrace{r + \gamma V^\kappa(s')}_{\text{target}} - Q^\pi(s, a) \right]. \quad (5.1)$$

Note that where in the standard Bellman equation we would have $V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a)$, in our case we have

$$V^\kappa(s) = \sum_{\sigma \in C} p(\sigma|s) \sum_a \sigma(s, a) Q^\pi(s, a), \quad (5.2)$$

computed as a weighted sum over all possible control policies $\sigma \in C$. Note that we can learn Q^π without actually experiencing any attack or malfunction, based only on prior assumptions about the possible control policies as captured by the operator κ . We refer to this target modification as the operator κ because it closely resembles the Bellman optimality operator \mathcal{T}^* , which is defined as $\mathcal{T}^*V(s) = \max_a [R(s, a) + \sum_{s'} T(s'|s, a) \gamma V(s')]$. Thus, we can then formally define the κ optimality operator \mathcal{T}_κ^* by substituting the value function $V(\cdot)$ with $V^\kappa(\cdot)$.

In Section 5.2.3 we present several κ -versions of classical TD methods. For simplicity we assume a scenario in which we have only a single adversarial external policy σ that aims to minimise our value, and thus $C = \{\pi, \sigma\}$. Note however that our model is general, and would work for any C and $p(\cdot)$. We now present a more general model assuming several Q-value functions, allowing for more complex control.

5.2.2 Advanced Model

We follow up on the model presented. We describe a more advanced model with two entities potentially having control but now the not-in-control policy cannot be defined in terms of our Q-value function Q^π and instead is based on a different Q-value function Q^σ . We approach this by learning separate Q-functions for both our target policy π and the not-in-control (e.g., attacker) policy σ , and using a mix of both Q-functions based on the control transition model (e.g., probability of attack) to evaluate the next state. The Q-update for our safe policy π is the same but the value function is now

$$V^\kappa(s) = \sum_{\sigma \in C} p(\sigma|s) \sum_a \sigma(s, a) Q^\sigma(s, a), \quad (5.3)$$

where the target is a weighted sum³ of several value functions learned using all different possible policies σ . Technically, we want to find our robust policy π by learning the Q-function Q^π while assuming that in the next state s' *any* control domain $\sigma_i \in C$ can be active with some probability $p(\sigma_i|s')$, where $\sum_\sigma p(\sigma_i|s') = 1$. Thus, at the same time of learning Q^π we want to learn Q-functions for all policies σ we consider. The value function of the not-in-control policy σ can be learned from the same experience stream $\langle s, a, r, s' \rangle$ generated by policy π by using importance sampling⁴ denoted by $c_s = \frac{\sigma(a|s)}{\pi(a|s)}$ as $Q^\sigma(s, a) \leftarrow Q^\sigma(s, a) + \alpha c_s [r(s, a) + \gamma V^\kappa(s') - Q^\sigma(s, a)]$. This gives us our own *estimation* of the Q-functions for all policies σ . In the remainder of this chapter we focus on the case with a single Q-value function as described in Section 5.2.1 and leave further analysis with the advanced model for future work.

5.2.3 Examples of TD(κ) Methods

We first present single-agent κ -based learning methods by building on the standard TD methods Q-learning and Expected SARSA. Then we present two-agent joint-action learning approaches. Although a generalisation to n agents is relatively straightforward, we choose to focus solely on the single- and two-agent case in this chapter for clarity of exposition. In each case, we consider the setting in which either the focal agent, with policy π , is in control, or the external adversary with policy σ aiming to minimise return. We further

³In general this can be any operator, not just *sum* (i.e., linear combination), which would allow for more complex scenarios.

⁴Or other methods similar to importance sampling like in Retrace(λ) algorithm [Munos et al., 2016].

simplify the model by making the control policy probability function $p(\cdot)$ state-independent, reducing it to a probability vector.

Single-Agent Methods

Before we present the algorithms, it is important to note that we need to distinguish the *target* and *behaviour* policies. The κ -operator is defined on the target (see Eq. (5.4)), while the behaviour policy is used only for selecting actions. We assume an ϵ -greedy behaviour policy throughout.

In **off-policy $Q(\kappa)$** , the target policy is the greedy policy $\pi(s) = \arg \max_a Q(s, a)$ that maximises expected return. The adversarial policy on the other hand aims to minimise the return, i.e., $\sigma(s) = \arg \min_a Q(s, a)$. Assuming a probability of attack of \varkappa as before, we have $p(\pi) = (1 - \varkappa)$ and $p(\sigma) = \varkappa$. Thus, Eq. (5.2) becomes

$$V^\kappa(s) = (1 - \varkappa) \max_a Q(s, a) + \varkappa \min_a Q(s, a).$$

For **on-policy Expected SARSA(κ)** the target is the (expectation over the) focal policy π , while the adversarial policy σ remains the same as before. Thus, we have

$$\begin{aligned} V^\kappa(s) &= (1 - \varkappa) \mathbb{E}_{a \sim \pi} [Q(s, a)] + \varkappa \min_a Q(s, a) \\ &= (1 - \varkappa) \sum_a \pi(a|s) Q(s, a) + \varkappa \min_a Q(s, a). \end{aligned}$$

Multi-Agent Methods

We move from a single-agent setting to a scenario in which multiple agents interact. For sake of exposition we only present a two-agent case (not counting the potential external control), which we further examine in the remainder of this chapter.

We assume two agents with different action spaces, A_1 and A_2 , but an identical reward function and thus a shared joint action Q-value function $Q : S \times A_1 \times A_2 \rightarrow \mathbb{R}$. Moreover, we assume full communication during the learning phase, allowing the agents to take each other's policies into account when selecting the next action.⁵ Our algorithms are therefore based on the joint-action learning (JAL) paradigm [Claus and Boutilier, 1998]. The Q-value

⁵A common practice in cooperative multi-agent learning settings, see e.g., [Foerster et al., 2018; Sunehag et al., 2018].

function update is thus defined as

$$Q^\pi(s, \langle a_1, a_2 \rangle) \leftarrow Q^\pi(s, \langle a_1, a_2 \rangle) + \alpha \left[r + \gamma V^\kappa(s') - Q^\pi(s, \langle a_1, a_2 \rangle) \right]. \quad (5.4)$$

We further assume that only one agent can be attacked at each time step.⁶ For **multi-agent** $\mathbf{Q}(\kappa)$ we can write Eq. (5.2) for each individual agent as

$$\begin{aligned} V^\kappa(s) &= (1 - \varkappa) \max_{A_1} \max_{A_2} Q(s, \langle a_1, a_2 \rangle) \\ &+ \frac{\varkappa}{2} \min_{A_1} \max_{A_2} Q(s, \langle a_1, a_2 \rangle) \\ &+ \frac{\varkappa}{2} \min_{A_2} \max_{A_1} Q(s, \langle a_1, a_2 \rangle) \end{aligned}$$

with $a_1 \in A_1$ and $a_2 \in A_2$, representing the scenario in which no attack happens with probability $(1 - \varkappa)$, and each agent is attacked individually with probability $\varkappa/2$.⁷ Analogously, we can define Eq. (5.2) for **multi-agent Expected SARSA**(κ) as

$$\begin{aligned} V^\kappa(s) &= (1 - \varkappa) \mathbb{E}_{a_1 \sim \pi_1, a_2 \sim \pi_2} [Q(s, \langle a_1, a_2 \rangle)] \\ &+ \frac{\varkappa}{2} \min_{A_1} \mathbb{E}_{a_2 \sim \pi_2} [Q(s, \langle a_1, a_2 \rangle)] \\ &+ \frac{\varkappa}{2} \min_{A_2} \mathbb{E}_{a_1 \sim \pi_1} [Q(s, \langle a_1, a_2 \rangle)], \end{aligned}$$

where we now compute an expectation over the actual policy of the agents that are not attacked, while the attacker is still minimising.

Another direction of multi-agent learning would be assuming no communication between the agents. Such assumption would further complicate the learning process due to inability of the agents to agree on a joint action. Each agent would need to learn an individual Q-value function and only best-respond to an estimated policy of the other agent(s). One could assume a simple policy estimation such as *fictitious play* as described in Section 2.3. This is an interesting extension of the proposed robust temporal difference learning operator to more general settings, nevertheless we leave this direction for future work.

⁶Although relaxing this assumption is straightforward, we opt to keep it for clarity.

⁷Note the order of the min max, which follows the Stackelberg assumption of an all-knowing attacker who moves last.

5.3 Theoretical Analysis of Convergence of Operator κ

In this section we analyse theoretical properties of the proposed κ -methods. We start by relating the different algorithms to each other in the limit of their respective parameters. Then we proceed to show convergence of both $Q(\kappa)$ and Expected SARSA(κ) to two different fixed points: (i) to the optimal value function Q^* of the original MDP in the limit where $\varkappa \rightarrow 0$; and (ii) to the optimal robust value function Q_κ^* of the MDP that is *generalised* w.r.t. κ for constant parameter \varkappa . Note that *optimality* in this sense is purely induced by the relevant operator. In (i) this is the standard Bellman optimality which maximises the expected discounted return of the MDP. However, in (ii) we derive optimality in the context of *Generalised MDPs* [Szepesvari and Littman, 1997], where optimal simply means the fixed point of a given operator, which can take many forms.

Before proceeding with the convergence proofs, Figure 5.1 summarises some relationships between the algorithms in terms of their targets, in the limit of their respective parameters: As is known, Expected SARSA, SARSA, and Q-learning become identical in the limit of a greedy policy [Sutton and Barto, 1998; Van Seijen et al., 2009]. Furthermore, the update targets of our κ -methods approach the update targets of the standard TD methods on which they are based as $\varkappa \rightarrow 0$. Finally, Expected SARSA(κ) and $Q(\kappa)$ share the same relationship as their original versions, and thus Expected SARSA(κ) approaches $Q(\kappa)$ as $\epsilon \rightarrow 0$. Note that the algorithms' equivalence in the limit does not hold in the transient phase of the learning process, and hence in practice they may converge on different paths and to different policies that share the same value function. For a comprehensive understanding of the algorithms introduced in Section 5.2.3, the following sections provide proofs for both convergence of κ methods for $\varkappa \rightarrow 0$, as well as their convergence when \varkappa stays constant.⁸

5.3.1 Convergence to the Optimal Q^*

There exist several proofs of convergence for the temporal difference algorithms Q-learning [Jaakkola et al., 1994; Tsitsiklis, 1994], SARSA [Singh et al., 2000], and Expected SARSA [Van Seijen et al., 2009]. Each of these proofs hinges on linking the studied algorithm to a stochastic process, and then using convergence results from stochastic approximation theory

⁸While we focus on the adversarial targets considered in Section 5.2.3, a previous proof of convergence under persistent exploration [Szepesvari and Littman, 1997] can be interpreted as a model of random failures with fixed kappa.

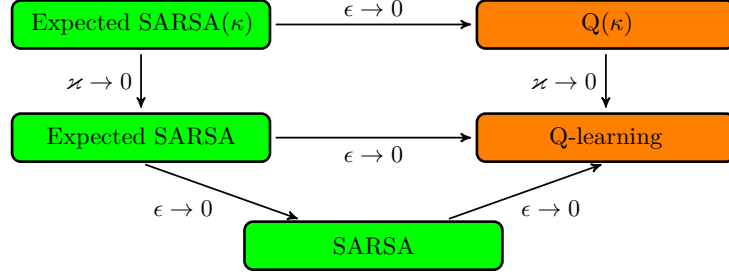


Figure 5.1: The relationship between the learning targets of different algorithms in the limits of their parameters. On-policy methods are in green, off-policy methods in orange.

[Dvoretzky, 1956; Robbins and Monro, 1951]. These proofs are based on the following lemma, presented as Theorem 1 in Jaakkola et al. [1994] and as Lemma 1 in Singh et al. [2000]. These differ in the third condition, which describes the contraction mapping of the operator. The contraction property used for the Q-learning proof [Jaakkola et al., 1994] has the form $\|\mathbb{E}\{F_t(\cdot)|P_t\}\| \leq \gamma\|\Delta_t\|$, where $\gamma \in [0, 1)$. We show the lemma as it was used for the SARSA proof provided by Singh et al. [2000], who show that the contraction property does not need to be strict; strict contraction is required to hold only asymptotically.⁹

Lemma 5.3.1. *Consider a stochastic process $(\alpha_t, \Delta_t, F_t), t \geq 0$, where $\alpha_t, \Delta_t, F_t : X \rightarrow \mathbb{R}$ satisfy the equations*

$$\Delta_{t+1}(x) = (1 - \alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x), x \in X, t = 0, 1, 2, \dots$$

Let P_t be a sequence of increasing σ -fields such that α_0 and Δ_0 are P_0 -measurable and α_t, Δ_t and F_{t-1} are P_t -measurable, $t = 1, 2, \dots$. Then, Δ_t converges to zero with probability one (w.p.1) under the following assumptions:

1. *the set X is finite,*
2. *$0 \leq \alpha_t(x_t) \leq 1$, $\sum_t \alpha_t(x_t) = \infty$, $\sum_t \alpha_t^2(x_t) < \infty$ w.p.1,*
3. *$\|\mathbb{E}\{F_t(\cdot)|P_t\}\| \leq \gamma\|\Delta_t\| + c_t$, where $\gamma \in [0, 1)$ and c_t converges to zero w.p.1,*
4. *$\text{Var}\{F_t(x_t)|P_t\} \leq K(1 + \|\Delta_t\|)^2$, where K is some constant,*

where $\|\cdot\|$ denotes a maximum norm.

⁹For proof based on Proposition 4.5 of Bertsekas [1995] see Appendix A of Singh et al. [2000].

The proof continues by relating Lemma 5.3.1 to the temporal difference algorithm, following the same reasoning as Van Seijen et al. [2009] in their convergence proof for Expected SARSA. We define $X = S \times A$, $P_t = \{Q_0, s_0, a_0, r_0, \alpha_0, s_1, a_1, \dots, s_t, a_t\}$, $x_t = (s_t, a_t)$, which represents the past at step t and $\alpha_t(x_t) = \alpha_t(s_t, a_t)$ is a learning rate for state s_t and action a_t . To show the convergence of Q to the optimal fixed point Q^* we set $\Delta_t(x_t) = Q_t(s_t, a_t) - Q^*(s_t, a_t)$, therefore when Δ_t converges to zero, then the Q values converge to Q^* . The maximum norm $\|\cdot\|$ can be expressed as maximising over states and actions as $\|\Delta_t\| = \max_s \max_a |Q_t(s, a) - Q^*(s, a)|$.

We follow the reasoning of Theorem 1 from Van Seijen et al. [2009], where we repeat the conditions (1), (2) and (4) and modify the condition **(3)** for the κ methods as:

Theorem 5.3.2. *$Q(\kappa)$ and Expected SARSA(κ) as defined in Section 5.2.3 using the respective value function V^κ , defined by*

$$Q_{t+1}(s_t, a_t) = (1 - \alpha_t(s_t, a_t))Q_t(s_t, a_t) + \alpha_t(s_t, a_t)[r_t + \gamma V_t^\kappa(s_{t+1})]$$

converge to the optimal Q function $Q^(s, a)$ if:*

1. *the state space S and action space A are finite,*
2. *$\alpha_t(s_t, a_t) \in (0, 1)$, $\sum_t \alpha_t(s_t, a_t) = \infty$ and $\sum_t \alpha_t^2(s_t, a_t) < \infty$ w.p.1,*
- (3)** *\varkappa converges to zero w.p.1,*
- (3a) for Expected SARSA(κ) the policy is greedy in the limit with infinite exploration (GLIE assumption),*
4. *the reward function is bounded.*

Proof. Convergence of $Q(\kappa)$: To prove convergence of $Q(\kappa)$ we have to show that the conditions from Lemma 5.3.1 hold. Conditions (1), (2) and (4) of Theorem 5.3.2 correspond to conditions (1), (2) and (4) of Lemma 5.3.1 [Van Seijen et al., 2009]. We now need to show that the contraction property holds as well, using condition (3) of Theorem 5.3.2. Adapting the proof of Van Seijen et al. [2009], we set $F_t(x) = F_t(s, a) = r_t(s, a) + \gamma V_t^\kappa(s') - Q^*(s, a)$ to show that $F_t(s, a)$ is a contraction mapping, i.e., condition (3) in Lemma 5.3.1. For $Q(\kappa)$ we write:

$$F_t = r_t + \gamma((1 - \varkappa) \max_a Q_t(s_{t+1}, a) + \varkappa \min_a Q_t(s_{t+1}, a)) - Q^*(s_t, a_t).$$

We want to show that $\|\mathbb{E}\{F_t\}\| \leq \gamma\|\Delta_t\| + c_t$ to prove the convergence of $Q(\kappa)$ to the optimal value Q^* .

$$\begin{aligned}
\|\mathbb{E}\{F_t\}\| &= \|\mathbb{E}\{r_t + \gamma((1 - \kappa) \max_a Q_t(s_{t+1}, a) + \kappa \min_a Q_t(s_{t+1}, a)) - Q^*(s_t, a_t)\}\| \\
&\leq \|\mathbb{E}\{r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q^*(s_t, a_t)\}\| + \\
&\quad \gamma \|\mathbb{E}\{\kappa \min_a Q_t(s_{t+1}, a) - \kappa \max_a Q_t(s_{t+1}, a)\}\| \\
&\leq \gamma \max_s |\max_a Q_t(s, a) - \max_a Q^*(s, a)| + \\
&\quad \gamma \max_s |\kappa \min_a Q_t(s, a) - \kappa \max_a Q_t(s, a)| \\
&\leq \gamma\|\Delta_t\| + \\
&\quad \gamma \kappa \max_s |\min_a Q_t(s, a) - \max_a Q_t(s, a)|,
\end{aligned}$$

where the first inequality follows from standard algebra and the fact that splitting the maximum norm yields at least as large a number, the second inequality follows from the definition of Q^{*10} and the maximal difference in values over all states being at least as large as a difference between values given in state s_{t+1} , and the third inequality follows from the definition of $\|\Delta_t\|$ above. We can see that if we set $c_t = \gamma \kappa \max_s |\min_a Q_t(s, a) - \max_a Q_t(s, a)|$, then for $\kappa \rightarrow 0$ we get c_t converging to zero w.p.1, thus proving convergence of $Q(\kappa)$. ■

Proof. Convergence of Expected SARSA(κ): Similarly as in the proof of $Q(\kappa)$ we need to show that the contraction property holds as well, this time using conditions (3) and (3a) of Theorem 5.3.2. We first define:

$$F_t = r_t + \gamma((1 - \kappa) \sum_a \pi_t(a|s_{t+1}) Q_t(s_{t+1}, a) + \kappa \min_a Q_t(s_{t+1}, a)) - Q^*(s_t, a_t)$$

¹⁰Recall that we set out in this section to show convergence to the same optimal Q-value as classical Q-learning $Q^*(s_t, a) = r_t + \gamma \max_{a'} Q^*(s_{t+1}, a')$, even if we do so by our new operator.

and then show the following:

$$\begin{aligned}
\|\mathbb{E}\{F_t\}\| &= \|\mathbb{E}\{r_t + \gamma((1 - \varkappa) \sum_a \pi_t(a|s_{t+1})Q_t(s_{t+1}, a) + \varkappa \min_a Q_t(s_{t+1}, a)) - Q^*(s_t, a_t)\}\| \\
&\leq \|\mathbb{E}\{r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q^*(s_t, a_t)\}\| + \\
&\quad \gamma \|\mathbb{E}\{(1 - \varkappa) \sum_a \pi_t(a|s_{t+1})Q_t(s_{t+1}, a) + \varkappa \min_a Q_t(s_{t+1}, a) - \max_a Q_t(s_{t+1}, a)\}\| \\
&\leq \gamma \max_s |\max_a Q_t(s, a) - \max_a Q^*(s, a)| + \\
&\quad \gamma \max_s |(1 - \varkappa) \sum_a \pi_t(a|s)Q_t(s, a) + \varkappa \min_a Q_t(s, a) - \max_a Q_t(s, a)|,
\end{aligned}$$

where the inequalities use the same operations as above in the proof of $Q(\kappa)$. If we set $c_t = \gamma \max_s |(1 - \varkappa) \sum_a \pi_t(a|s)Q_t(s, a) + \varkappa \min_a Q_t(s, a) - \max_a Q_t(s, a)|$ and assume that the policy is greedy in the limit with infinite exploration (GLIE assumption) and parameter $\varkappa \rightarrow 0$ w.p.1 (conditions (3) and (3a)), it follows that c_t converges to zero w.p.1, thereby proving that Expected SARSA(κ) converges to optimal fixed point Q^* . ■

5.3.2 Convergence to the Robust Q_κ^*

In this section we show convergence to the robust value function Q_κ^* which is optimal w.r.t. the operator κ . The main difference with the proof of Theorem 5.3.2 is that here we do not require $\varkappa \rightarrow 0$ but instead assume it remains constant over time. We base our reasoning on the theory of *Generalised MDPs* [Szepesvari and Littman, 1997]. A Generalised MDP is defined using operator-based notation as

$$\left(\bigotimes \bigoplus (R + \gamma V) \right)(s) = \max_a \sum_{s'} T(s'|s, a) (R(s, a) + \gamma V(s')),$$

where the operator \bigotimes defines how an optimal agent chooses his actions (in the classic Bellman equation this denotes maximisation) and operator \bigoplus defines how the value of the current state is updated by the value of the next state (in the classic Bellman equation this denotes a probability weighted average over the transition function). These operators can be chosen to model various different scenarios. The generalised Bellman equation can now be written as $V^* = \bigotimes \bigoplus (R + \gamma V^*)$. The main result of Szepesvari and Littman [1997] is that if \bigotimes and \bigoplus are non-expansions, then there is a unique optimal solution to which the generalised Bellman equation converges, given certain assumptions. For $0 \leq \gamma < 1$

and non-expansion properties of \otimes and \oplus we get a contraction mapping of the Bellman operator \mathcal{T} defined as $\mathcal{T}V = \otimes \oplus (R + \gamma V)$. Then, the operator \mathcal{T} has a unique fixed point by the Banach fixed-point theorem [Smart, 1974].

Building on the stochastic approximation theory results (as we also used in the Section 5.3.1), Szepesvari and Littman [1997] show the following:

Lemma 5.3.3. *Generalised Q-learning with operator \otimes using Bellman operator*

$$\mathcal{T}_t(Q', Q)(s, a) = \begin{cases} (1 - \alpha_t(s, a))Q'(s, a) + \alpha_t(s, a)(r_t + \gamma(\otimes Q)(s'_t)) & \text{if } s = s_t, a = a_t \\ Q'(s, a) & \text{otherwise} \end{cases}$$

converges to the optimal Q function w.p.1, if

1. s'_t is randomly selected according to the probability distribution defined by $T(s_t, a_t, \cdot)$,
2. $\alpha_t(s_t, a_t) \in (0, 1)$, $\sum_t \alpha_t(s_t, a_t) = \infty$ and $\sum_t \alpha_t^2(s_t, a_t) < \infty$ w.p.1,
3. \otimes is a non-expansion,
4. the reward function is bounded.

We base our convergence proofs for $Q(\kappa)$ and Expected SARSA(κ) on the insights of Szepesvari and Littman [1997] given in Lemma 5.3.3.

Theorem 5.3.4. *$Q(\kappa)$ and Expected SARSA(κ) as defined in Section 5.2.3 converge to the robust Q function Q_κ^* for any fixed \varkappa .*

Proof. Convergence of $Q(\kappa)$ to Q_κ^* : To prove convergence of $Q(\kappa)$ we follow the proof of Generalised Q-learning in Lemma 5.3.3. The only condition we need to guarantee is the non-expansion property of the operator in the value function update, which for $Q(\kappa)$ is a weighted average of the operators *min* and *max*. We write the operator \otimes for $Q(\kappa)$ as \otimes^κ and define it as

$$(\otimes^\kappa Q)(s, a) = (1 - \varkappa) \max_a Q(s, a) + \varkappa \min_a Q(s, a).$$

In Appendix B of Szepesvari and Littman [1997], Theorem 9 states that any linear combination of non-expansion operators is also a non-expansion operator. Moreover Theorem 8 states that the summary operators *max* and *min* are also non-expansions. Therefore, \otimes^κ

is a non-expansion as well, thus proving the convergence of $Q(\kappa)$ to the robust fixed point Q_κ^* induced by the operator κ . ■

Proof. Convergence of Expected SARSA(κ) to Q_κ^* : We base our convergence proof of Expected SARSA(κ) again on the work of Szepesvari and Littman [1997], this time on their insights regarding persistent exploration (Section 4.5 in their paper). They show that Generalised Q-learning with ϵ -greedy action selection converges, for a fixed ϵ , in the Generalised MDP. Following similar reasoning, we define the operator \otimes for Expected SARSA(κ) with fixed ϵ as

$$(\otimes^\kappa Q)(s, a) = (1 - \varkappa) \left(\epsilon \frac{1}{|A|} \sum_a Q(s, a) + (1 - \epsilon) \max_a Q(s, a) \right) + \varkappa \min_a Q(s, a).$$

Again, from repeated application of Theorems 8 and 9 in Appendix B of Szepesvari and Littman [1997] it follows that \otimes^κ is a non-expansion as well. Therefore, by Lemma 5.3.3, Expected SARSA(κ) converges to Q_κ^* for fixed exploration ϵ . ■

It remains an open question whether Expected SARSA(κ) also converges for decreasing ϵ , e.g., under the GLIE assumption, even though we conjecture that it might.

5.3.3 Convergence in the Multi-Agent Case

We now prove convergence of the cooperative multi-agent variant of the κ methods presented in Section 5.2.3. This proof builds on the theory of Generalised MDPs, similar to the proofs presented in Section 5.3.2. Therefore this proof also assumes a fixed probability of attack \varkappa . In addition, we make use of the assumption that agents can communicate freely in the learning phase, and thus receive identical information and can build a common joint-action Q-table.

Theorem 5.3.5. *Multi-agent $Q(\kappa)$ and Expected SARSA(κ) as defined in Section 5.2.3 converge to the robust Q function Q_κ^* for any fixed \varkappa .*

Proof. The \otimes^κ operator for our multi-agent versions of $Q(\kappa)$ and Expected SARSA(κ) consists of a nested combination of different components, in particular $\max_a Q(s, a)$, $\min_a Q(s, a)$, and $\sum_a \pi^\epsilon(s, a) Q(s, a)$ where π^ϵ is the ϵ -greedy policy. By Theorem 8 of Szepesvari and Littman [1997], max and min are non-expansions. By Theorem 9 of Szepesvari and Littman [1997], linear combinations of non-expansion operators are also non-expansion operators. Finally, by Theorem 10 of Szepesvari and Littman [1997], products

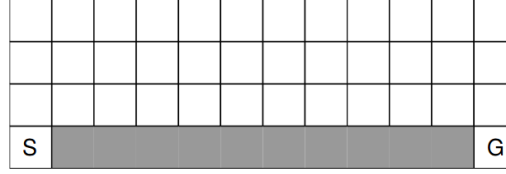


Figure 5.2: Cliff Walking: The agent needs to get from the start [S] to the goal [G], avoiding the cliff (grey tiles).

of non-expansion operators are also non-expansion operators. Therefore, also $\max \max$, $\max \min$, and $\min \max$ are non-expansion operators, as are linear combinations of those compounds. Similarly, $\sum_a \pi^\epsilon(s, a)Q(s, a)$ for fixed ϵ can be written as a linear combination of summary operators, which by Theorems 8 and 9 of Szepesvari and Littman [1997] is a non-expansion. Therefore, the \bigotimes^κ operator used in both multi-agent $Q(\kappa)$ and Expected SARSA(κ) is a non-expansion. Thus, by Lemma 5.3.3, $Q(\kappa)$ and Expected SARSA(κ) converge to Q_κ^* for fixed κ , and in the case of Expected SARSA(κ), for fixed ϵ . ■

5.4 Experiments with Robust Learning

In this section we evaluate temporal difference methods with the proposed operator κ : off-policy type of learning $Q(\kappa)$ and on-policy type of learning Expected SARSA(κ). We experiment with a classic cliff walking scenario for the single-agent case and a multi-agent puddle world scenario. Both these domains contain some critical states, a cliff and a puddle respectively, which render very high negative reward for the agent(s) in case of stepping into them. These critical states represent the significant rare events (SREs). We compare our methods to classic temporal difference methods like SARSA, Q-learning and Expected SARSA. In all the experiments we consider an undiscounted ($\gamma=1$) episodic scenario.

Cliff Walking: single-agent The Cliff Walking experiment as shown in Figure 5.2 is a classical scenario proposed in Sutton and Barto [1998] and used in many other papers ever since (e.g., Van Seijen et al. [2009]). The agent needs to get from the start state [S] to the goal state [G], while avoiding stepping into the cliff, otherwise rendering a reward of -100 and sending him back to the start. For every move which does not lead into the cliff the agent receives a reward of -1.

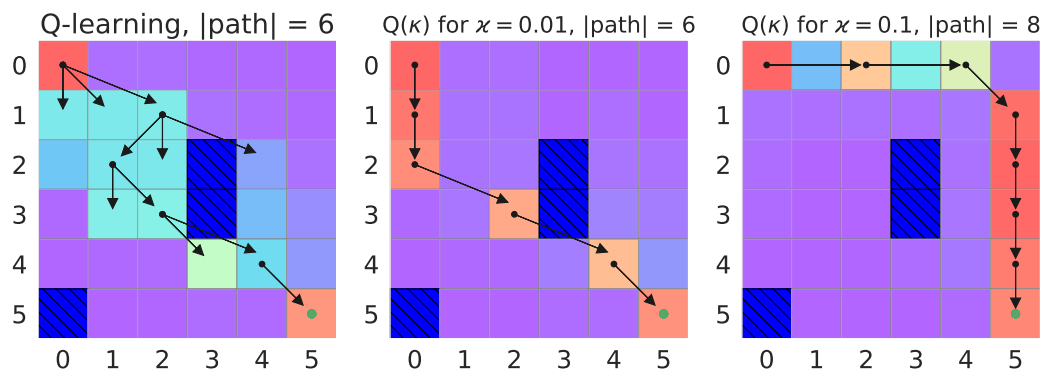


Figure 5.3: The Puddle World: $Q(\kappa)$ learns a safer path with increasing κ . Puddles are dark blue, the arrows show the optimal actions on the learned path, and the heatmap shows the number of visits to each state (red to blue, blue is none).

Puddle World: multi-agent The Puddle World environment is a grid world with puddles which need to be avoided by the joint-action learning agents. The two agents jointly control the movement of a single robot in the Puddle World, each controlling either direction $\langle \text{up}, \text{down} \rangle$ or $\langle \text{left}, \text{right} \rangle$. Agent 1 can take the actions $\{\text{stay}, \text{move down}, \text{move up}\}$ and agent 2 can choose $\{\text{stay}, \text{move left}, \text{move right}, \text{move right by 2}\}$, thus their action spaces are different, further complicating the learning process compared to the single-agent scenario. The joint action is the combination of the two selected actions. We assume a reward of -1 for every move and -100 for stepping into a puddle (returning to the start node). The agents have to move together from the start node at the top left corner to the goal at the bottom right corner. Figure 5.3 shows the policy learned by our proposed algorithm $Q(\kappa)$ for the two joint-learning agents. Note how a safer path (longer, avoiding the puddles) is learned with increasing parameter κ (i.e., higher probability of SREs). For $\kappa = 0$ our algorithm degenerates to Q-learning (left panel).

5.4.1 Performance

We replicate the experiment of Van Seijen et al. [2009] on the Cliff Walking domain, in which we compare our κ methods with Q-learning, SARSA and Expected SARSA, and perform a similar experiment on the Puddle World domain. In line with Van Seijen et al. [2009] we show (i) early performance, which is the average return over the first 100 training episodes, and (ii) converged performance, which is the average return over 100,000 episodes.

Figure 5.4 shows the results for three different settings of both scenarios: (i) a determin-

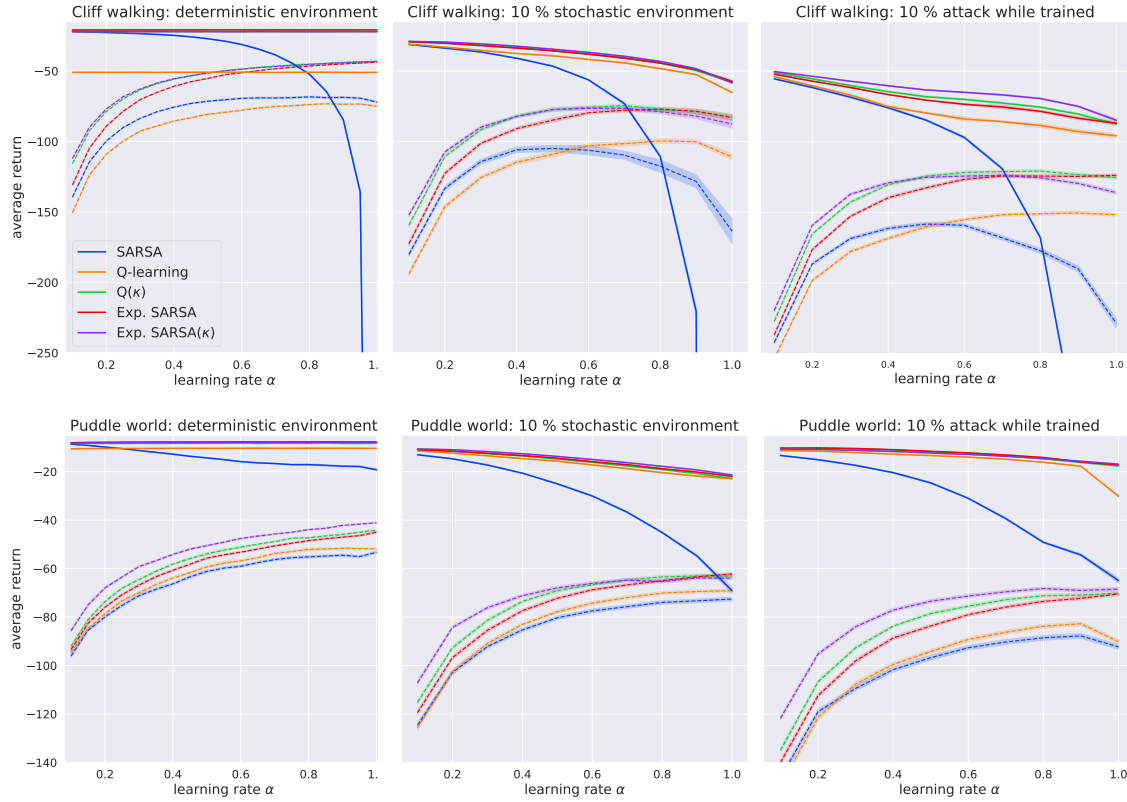


Figure 5.4: Cliff Walking (single-agent) in first row and Puddle World (multi-agent) in second row. Deterministic environment (first column), 10 % stochastic environment (second column) and 10 % attack while training (third column). ϵ -greedy policy with fixed $\epsilon = 0.1$. Early performance - dashed lines (100 episodes), converged performance - solid lines (100,000 episodes).

istic environment, where each action chosen by the policy is executed with certainty; (ii) an environment with 10% stochasticity, in which a random action is taken 10% of the time; and (iii) an environment with 10% probability of attack, in which an adversarial action is taken 10% of the time. As before, we define an attack as an action that minimises the Q-value in the given state. The stochastic environment can be seen as modelling random failures.

The early performance experiments are averaged over 300 trials and the converged performance experiments are averaged over 10 trials. We also show the 95% confidence intervals on all results. We fix the exploration rate to $\epsilon = 0.1$; for the κ methods we set $\varkappa = 0.1$ (later in this section we also experiment with different settings of \varkappa). Note that the y-axis, showing the average return, is the same in each row for easy comparison. The x-axis shows different learning rates α . We can see how the average return decreases with more complex scenarios, from deterministic, over to stochastic, to one with attacks. The κ methods are superior to the other baselines in the early performance experiments, especially in the attack case, which is the scenario the κ methods are designed for. In the converged performance experiments the κ methods beat Q-learning and SARSA and perform at least as well as Expected SARSA.

5.4.2 Different Levels of Attack

In this section we investigate how the methods behave under different levels of attack, defined by the probability of attack per state. We consider an attack on trained (converged) methods, thus we first train each method for 100,000 episodes (in deterministic environment) and then we test it on 50,000 trials with given probability of attack per state. We average the results over 10 trials and provide 95% confidence intervals. Note, that this is a different methodology of testing the methods against an adversarial attack compared to the experiments in Figure 5.4, where we considered attacks during training. This experiment shows the strength of the κ methods for different levels of attacks. We assume the probability of attack to be known here and thus we set the parameter \varkappa to be equal to that probability, which is the meaning of the parameter \varkappa as described before. In other words, parameter \varkappa prescribes how much safely we want to act. We consider very rare attacks (0.001 probability of attack in each state) to more frequent attacks (0.2 probability of attack in each state) as shown in Figure 5.5. For better visualisation we use logarithmic axes. We train all the methods with fixed exploration rate $\epsilon = 0.1$ and learning rate $\alpha = 0.1$, note that the

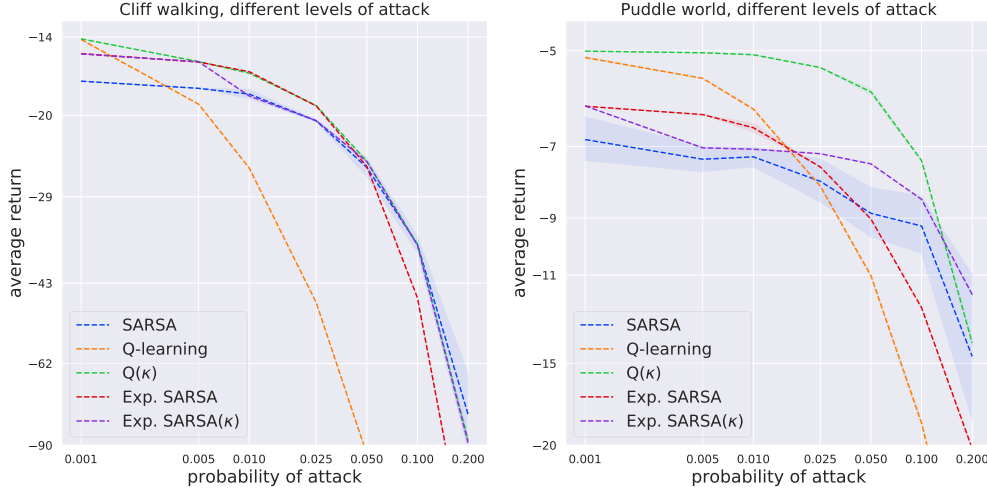


Figure 5.5: Varying probability of attack: Cliff Walking (left), Puddle World (right), trained on 100k episodes, tested on 50k episodes, $\alpha = 0.1$, $\epsilon = 0.1$.

methods (except SARSA) converge to the same result for different learning rates as shown in left panel of Figure 5.4. SARSA is very unstable for different learning rates (demonstrated by wide confidence intervals), learns different paths for different α and does not converge fast enough or not at all, which can be partly explained by its higher variance [Van Seijen et al., 2009]. We test the different levels of probability of attack on the Cliff Walking experiment in the left panel of Figure 5.5, where we can see that the κ methods compare favourably to the other baselines, however in some parts they give similar performance as Expected SARSA or SARSA. The Cliff Walking experiment has a limited expressiveness for testing the methods due to a limited number of possible safe paths with low costs (see Figure 5.2), which is the reason for the κ methods to show only similar performance compared to the baselines, not reaching their full potential. However, the Puddle World is more expressive, because there are several possible paths differing in level of safety and cost. The bigger solution space of the Puddle World is also induced by the two cooperating agents, each having their own action space. Therefore, on the right panel of Figure 5.5 we show the Puddle World experiment for different levels of probability of attack. Here, we can clearly see the κ methods outperform the baselines, especially $Q(\kappa)$ is superior over the whole range of considered probabilities of attack. Note that $Q(\kappa)$ learns a safer path even for very rare attacks (0.001), which is also shown in Figure 5.3, where $Q(\kappa)$ learns a path

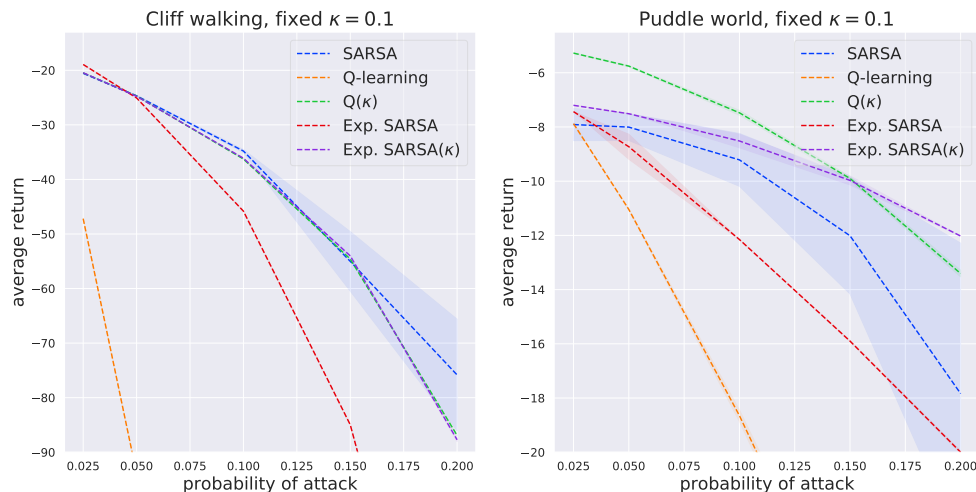


Figure 5.6: Robustness analysis: Cliff Walking (left), Puddle World (right), trained on 100k episodes, tested on 50k episodes, $\alpha = 0.1$, $\epsilon = 0.1$, $\varkappa = 0.1$.

with the same cost (distance) compared to Q-learning, but further from the puddles.

5.4.3 Robustness Analysis

We now test the robustness of the proposed algorithms to an incorrect attack model, meaning that the value of \varkappa in $Q(\kappa)$ and Expected SARSA(κ) no longer matches the actual probability of attack (in our previous experiments \varkappa matched the actual probability of attack precisely). Figure 5.6 shows the performance of our algorithms for a range of actual attack probabilities (x-axis) while learning using a fixed parameter $\varkappa = 0.1$.

To better highlight the robustness of our methods we choose a range of relatively high actual probabilities of attack around the fixed value of $\varkappa = 0.1$ (note that we no longer use a logarithmic scale). One can see that even when \varkappa is not equal to the actual probability of attack the proposed κ algorithms still outperform the baselines in most cases. In the Cliff Walking experiment (Figure 5.6 left) the κ methods perform similar to SARSA; however SARSA is quite unstable, as discussed before and as one can see by the width of the confidence interval. The Puddle World experiment (Figure 5.6 right) demonstrates the superior performance of the κ methods, which beat all the baselines even for the fixed parameter \varkappa . These results show that even when we do not know the probability of attack accurately we can learn a more robust strategy using the κ methods.

5.5 Discussion

We presented a new operator κ for temporal difference learning, which improves robustness of the learning process against potential attacks or perturbations in control. We proved convergence of $Q(\kappa)$ and Expected SARSA(κ) to: (i) the optimal value function Q^* of the original MDP in the limit where $\varkappa \rightarrow 0$; and (ii) the optimal robust value function Q_κ^* of the MDP that is generalised w.r.t. the operator κ for constant parameter \varkappa . In the latter case we also proved convergence of a cooperative joint-action learning version of our methods.

Our complementary empirical results demonstrate that the proposed κ -methods indeed provide robustness against a chosen scenario of potential attacks and failures in both single- and multi-agent settings. Although our method assumes that a model of such attacks and failures is known to the agent, we further demonstrate that our methods are robust against small model errors. Moreover, we show that even in absence of attacks or failures, our method learns a policy that is robust in general against environment stochasticity, in particular in the early stages of learning.

Our proposed operator κ can be closely linked or even combined with some recent state-of-the-art reinforcement learning methods. Considering a multi-step update, the operator could be combined with Retrace(λ) [Munos et al., 2016], which would potentially speed up convergence. Another promising extension of our model would be to combine it with $Q(\sigma)$ [De Asis et al., 2018] to allow for mixed multi-step updates. Note that the parameter σ in this algorithm can also be time- or state-dependent similarly to the potential extensions of the control in our model. This would allow to learn robust policies against more complex controls such as multi-step attacks, with a potentially non-uniform probability distribution of the control policies. Whereas our formal model assumes a uniform probability distribution of the control policies over the states and time steps. Another application along this line would be to model the control transition similar to the options framework [Sutton et al., 1999; Bacon et al., 2017], in which case the alternate control policies could be seen as “malicious” options over which the agent has no control, with potentially complex initiation sets and termination conditions. Furthermore, the target of adversarial policies could be learned from experience, where ideas from opponent modelling could be used (e.g., DPIQN [Hong et al., 2018]). Such extensions would further increase the flexibility of our proposed operator, making it applicable to a wide range of real-world scenarios.

This new family of temporal difference learning methods paves a new path to robust learning in systems with critical states leading to increased safety. We especially focus on systems where a compromise of individual nodes in a network can lead to severe impact on the whole system. This contribution further narrows the reality gap between theory and practise and potentially helps to mitigate some of the security threats such as external attacks or random failures in critical systems such as energy smart grids.

6

Learning When Facing Adversarial Agents in Spatial Security Domains

This chapter is based on the following publications:

- Klima, R., Tuyls, K., and Oliehoek, F. (2016c). Markov security games: Learning in spatial security problems. *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*, pages 1–8
- Klima, R., Tuyls, K., and Oliehoek, F. (2018d). Model-based reinforcement learning under periodical observability. *AAAI Spring Symposium on Learning, Inference, and Control of Multi-Agent Systems*

6.1 Uncertainty in Spatial Security Domains

In this chapter we tackle the challenge of modelling adversarial agents in order to learn effective strategies. We focus on security and sustainability in *spatial security problems*, briefly introduced in Section 3.1, which are problems defined between two (groups of) agents with opposing goals and who move on a map. We call these two (groups of) agents *the defender* and *the attacker*, where the main goal of the defender is to apprehend the attacker and the main goal of the attacker is to attack some critical targets in the map. There are several examples of such spatial security problems like terrorism in big cities [Jain et al., 2011], illegal rhino poaching or over-fishing [Fang et al., 2015]. The last two examples are prime instances of sustainability problems, where the lack of effective strategies to prevent the adversaries from attacking can lead to an extinction of the whole species. In this chapter we study the example domain of illegal rhino poaching to demonstrate our approach to spatial security games. In the illegal rhino poaching problem [Montesh, 2013] there are two groups of agents: rangers (the defender), who aim to secure and sustain the wildlife reservation and prevent rhinos from being poached; and poachers (the attacker), who try to poach the animals and make profit from selling those on a black market. Our proposed method is applicable to other spatial security game scenarios, where the general problem belongs to a domain of pursuit-evasion games, with the defender pursuing the attacker.

The high level goal of any effective behaviour in security domains is to effectively allocate limited resources of the defender to mitigate the threats. In this chapter we focus on deriving an effective strategy for the defender to move on the map to apprehend the attacker. What makes this task especially difficult is partial observability of the attacker, bringing severe uncertainty into the defender decision making. Effective modelling of such uncertainties is one of the main challenges in order to deploy AI techniques in real-world applications of security games. The reasoning agent often has access to extra information about the environment which if used properly can help significantly in effective strategy-making. In security games this knowledge can come from several types of surveillance available to the agent. In this thesis we investigate a model-based approach, where we continually learn and improve our model of the opponent behaviour. The main uncertainty lies in not being able to always observe the opponent's location. To tackle this challenge we develop a statistical probability model to enable us to reason about the opponent's location. We base opponent location modelling on the observed frequencies of transitions between states, defined as the location of both the defender and the attacker, and a prior information about the

environment e.g., target location.

The main research body in security games focuses on computing exact solutions and describing their theoretical properties, mostly targeting the equilibria concepts, e.g., Nash equilibria or Stackelberg equilibria [Korzhyk et al., 2011]. This line of research has been an important theoretical building stone for tackling the problem, however, these methods are sometimes difficult to apply to real world settings due to often unrealistic assumptions and simplifications. A different approach from computing an exact solution is to learn the strategy from interacting with the environment and using techniques from reinforcement learning. This approach helps to overcome some of the unrealistic assumptions made by the theoretical approaches. Hence, we model this domain of security games as a reward-based system, where the agents obtain rewards and thus can learn strategies on a trial-error basis. We show how the problem can be approached by methods from temporal difference reinforcement learning, based on a framework of stochastic games defined on top of a Markov decision process (MDP) as formally described in Section 2.3.1 and Section 2.2.1, respectively. In security domains we often face a strategic and intelligent attacker who is able to partially or fully observe the defender strategy, which introduces an information asymmetry to the players' knowledge about each other. This asymmetry is called the Stackelberg assumption or the Stackelberg attacker. This is a different assumption to the classic normal-form game where the optimal solution is a Nash equilibrium, in Stackelberg games the optimal solution is a Stackelberg equilibrium [Korzhyk et al., 2011]. We approach the problem of obtaining effective strategies by learning from interaction instead. One of the main uncertainties about the attacker is his location in the map, which might not be observable or only partially observable by the defender. We focus on a special case of partial (limited) observability, where we sometimes get to fully observe the attacker location. We consider periodical observability, which is inspired by the board game *Scotland Yard* where the player gets to observe the opponent location periodically, e.g., every 3 time steps. We also consider random full observation of the attacker location given by some probability in each time step. We claim that this type of observability is quite common in security domains where the defender gets to observe an opponent location by obtaining some extra information. For instance in the green security game scenarios like the rhino poaching problem, the rangers can be informed by the villagers living nearby about the current location of the poachers, or this information can also come from surveillance by drones [Montesh, 2013]. Our main goal is to make use of the extra information about the attacker location in reinforcement learning style methods, obtaining an adaptive strategy,

maximising the probability of apprehending the attacker. Our proposed algorithm is based on the Replicated Q-learning [Littman et al., 1995] algorithm, which combines the standard Q-learning with belief states in partially observable domains. We extend this algorithm with learning the model of the opponent’s behaviour by using Bayesian inference over observed state transitions and a prior information about the environment.

Related Work Our work can be described in terms of the taxonomy proposed by Hernandez-Leal et al. [2017], where the authors proposed classification of multi-agent learning algorithms in terms of environment observability, opponent adaptation capabilities and how the agent deals with non-stationarity. We assume observability of the defender’s local reward and partial observability of opponent’s actions. The opponent is assumed to change his strategy only within some bounds, restricting his behaviour from abrupt and drastic changes. This is explained by the notion of bounded rationality, which is often assumed in security games [Pita et al., 2010]. Such an assumption allows us to learn a fairly stable model of the opponent behaviour based on the (partially) observed past behaviour, which we can then use to form an effective defender strategy.

In the studied problem we face two difficulties: (i) how to learn a policy in an unknown environment, which we tackle by a reinforcement learning style method; and (ii) how to deal with partial observability of the environment, which we tackle by learning a model of the environment dynamics. In order to build the model of the environment dynamics represented by the transition function, we make use of Bayesian inference to link a priori information with the current information from observations. Therefore, related work studies Bayesian reinforcement learning. The domain of Bayesian reinforcement learning can be divided into probabilistic modelling of the transition function, value function, reward function or policy. For example, Dearden et al. [1998] proposed Bayesian Q-learning, which models the reward as a Normal distribution to update the value function in Q-learning and proposed a Bayesian approach to implicitly trade-off between exploration and exploitation in action selection. In this chapter we focus on probabilistic modelling of the transition function, where we also propose a combination of Bayesian approach and Q-learning, however in a substantially different way. Our method uses a Bayesian approach to model the transition function to derive belief states, by modelling and learning the partially observable attacker behaviour. Secondly to model the limited observability of the underlying MDP, we use some notions from the framework of partially observable Markov decision processes (POMDP) (see Section 2.2.4), which is a generalisation of the original MDP. We consider beliefs about

the states as probabilities of the system to be in a particular underlying state. In practice, it is often computationally intractable to solve bigger POMDPs exactly due to infinite number of states and approximations need to be introduced [Silver and Veness, 2010]. Furthermore, systems modelled as POMDPs often assume knowledge of the model of the environment, i.e., the transition, the reward and the observation function. In such a case, given the model, we can solve the POMDP, also called *planning*. However, knowing the model of the environment exactly is often an unrealistic assumption in many practical applications. One approach to relax this assumption is the model of Bayes-adaptive POMDP (BA-POMDP) proposed by Ross et al. [2007], which uses the Bayesian rule to derive the transition and observation functions from count vectors of transitions and observations. Katt et al. [2017] extend that and propose a more efficient learning approach BA-POMCP, using Monte Carlo planning. These models (POMDP and BA-POMDP) often assume that the underlying state of the original MDP cannot be ever fully observed. In our work we consider a special case where we can occasionally fully observe the underlying state and use this to compute the beliefs of succeeding states. Thus, we assume an occasional access to the original MDP. The observations consist of occasional full observation of the underlying state and the knowledge about the current information set defining admissible states the system is currently in. Hence, we do not need to explicitly learn or estimate the observation function $O(o|s', a)$, which is often necessary in BA-POMDP or POMDP models.

Fang et al. [2015] define the model of *green security games* to approach security games concerning sustainability such as the problem of illegal rhino poaching or over-fishing, which is based on the framework of *Stackelberg security games* [Kiekintveld et al., 2009]. Additionally related work using the framework of security games and focusing on the learning approach has studied the border patrol problem [Klima et al., 2014, 2015]. Some of these security games however, do not consider space or time, i.e., the time it takes the defender to travel to the target node, as part of the model. Spatial security games are also often modelled as extensive-form games [Korzhyk et al., 2011], where there has been lot of work proposing computing optimal strategies online or offline, especially for zero-sum games [Bosansky et al., 2016; Jain et al., 2011]. In contrast, we make use of a more expressive framework of stochastic games to model the environment, however use the notion of *information set* from extensive-form game theory. An et al. [2012] compute the optimal defender strategy to a learning attacker who can only partially observe the defender and updates his beliefs using a Dirichlet distribution, whereas in our work we assume the Stackelberg attacker who can fully observe the defender past moves and best respond to

it. Ganzfried and Sandholm [2011] present opponent modelling based on a Dirichlet prior distribution, where they combine a precomputed equilibrium strategy with observations of the attacker’s past moves. In contrast we focus on combining learning of the strategy with observations of the attacker’s past moves.

6.2 Partially Observable Model of Spatial Security Games

We study the problem of effective decision making in spatial security games, which are played on a graph with two non-cooperative (groups of) players with opposing (not necessarily strictly, assuming general-sum game) goals. We define these two (groups of) players as the defender and the attacker following the Stackelberg security game framework as described in Section 2.1.4. The model is inspired by the green security game framework where we are motivated by the problem of illegal rhino poaching, where the two groups of players are rangers (the defender) who try to apprehend illegal rhino poachers (the attacker) and thus protect rhinos from being poached. The rhinos represent the *targets* as a commonly used notion in security games for critical assets which need to be secured. The environment can be, for example, a wildlife reservation, which can be modelled as a graph (grid); see Figure 6.1 for an example of such a model.

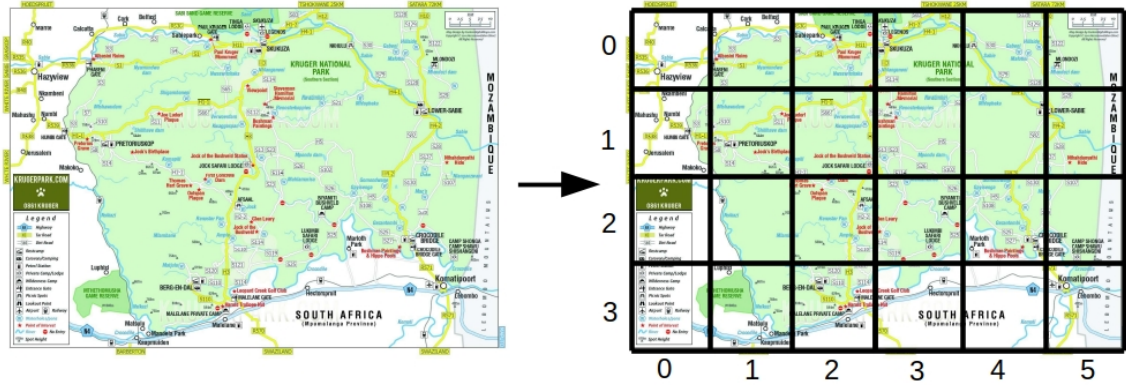


Figure 6.1: Example of modelling a wildlife reservation (Kruger park in South Africa) as a grid world. [www.safari.com/kruger-national-park/maps/kruger-park-far-south-section].

We model this problem as a stochastic game, using the notion of belief states as known in partially observable Markov decision processes. A state s is defined as a combination

of locations of the defender and the attacker in the grid; the action space A_d and A_a for the defender and the attacker, respectively, are identical for both of the players, allowing moving from one place in the grid to another neighbouring place. The defender reward function $R_d(s, \mathbf{a})$ is defined as a positive signal for the defender when apprehending the attacker (i.e., both players being in the same tile in the grid) and the attacker reward function $R_a(s, \mathbf{a})$ is a positive signal for successfully attacking the target (i.e., being in the same tile as the target). If both happen simultaneously then the apprehension has a priority and the attack is not successful. In our setting the underlying state is not always observable so we need to maintain a belief over each state $b(s)$ as a probability of the system being in that state.

6.2.1 Observability in Spatial Security Games

In the studied spatial security game we assume that the defender can always observe his own location in the map, but is uncertain about the attacker location. Our assumption is that the defender gets to fully observe the attacker location in some time steps, either periodically (i.e., every fixed number of steps) or with some probability in each time step. For example in the illegal poaching scenario this can be information about the attacker location from people living in the area or from surveillance by drones [Montesh, 2013]. The inability to always observe the attacker location means that the defender can always observe only his own location but not always fully observe the underlying state. However in general we assume that any state can be sometimes observed.

Therefore, the defender needs to maintain beliefs $b(s)$ over states which assign probabilities of being in each of the states s . This belief can also be expressed as the probability of an attacker being in a certain location given the location of the defender. We form these beliefs in each time step on a restricted state space of currently admissible states. We call this restricted set the *information set* (IS) for a given state and action. Generally speaking, knowing the state and action we can derive the next possible states given the structure of the grid (map). In case we cannot fully observe the state we can still use the current information set (states with non-zero beliefs) to form the succeeding information set (IS') of admissible states. In Figure 6.2 we show an example of a small grid world and corresponding extensive-form game tree with information sets. The defender is unsure about the current state, it is either $s_{1,5}$ or $s_{1,7}$, because the defender can always observe his own location (i.e., tile 1) but is unsure about the attacker location (either tile 5 or tile 7). The figure captures

a decision point, where the defender decides to go *down* (D). The defender reasons about the possible attacker actions and thus about the resulting attacker location in order to form the succeeding information set, which he derives to be $IS' = \{s_{2,2}; s_{2,4}; s_{2,6}; s_{2,8}\}$. States outside the information set have a zero belief, i.e., zero probability of the system being in that state. Note that in our analysis we use the model of a stochastic game instead of the model of an extensive-form game, due to higher expressibility and the possibility to transition between different stage games.

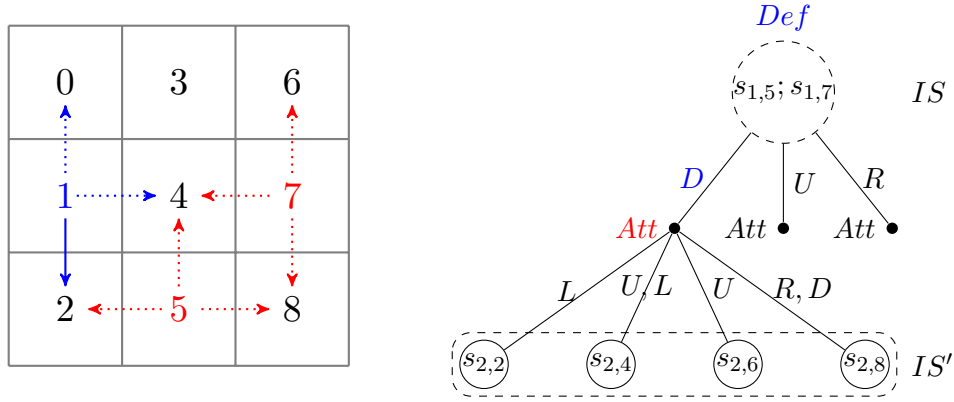


Figure 6.2: Example of states in information sets for the case where neither the current state nor the succeeding state is fully observed. We need to reason over two information sets; the current one IS and the succeeding one IS' . The defender is in location 1 and chooses the action *down* (D), the attacker is either in location 5 or 7.

Attacker behaviour model

In security domains we often face an adversarial opponent who is potentially intelligent and can observe to some extent the defender behaviour and plan his strategy accordingly, i.e., the Stackelberg assumption (see Section 2.1.4). Our assumption is that the attacker can observe all the past moves of the defender, which is a rather strong assumption but demonstrates the worst-case scenario in security games. We thus consider the attacker to follow a simple learning policy, best responding to the empirical frequencies of defender's past moves, which is a version of fictitious play (FP), described in Section 2.3.2. In our model, the attacker finds the shortest paths from the start node to the targets which are weighted by the defender visits in each node in the graph, i.e., the attacker prefers the tiles less visited by the defender. Hence, in every episode the attacker chooses the safest path to

the least risky target, minimising the probability of getting apprehended.

6.2.2 Statistical Approach to Uncertainty

We assume the defender to be uncertain about the location and the strategy of the opponent. Our main goal is to act efficiently under such uncertainty. In security games the defender often has access to some extra information about the attacker's whereabouts, which we use to deal with this uncertainty. We approach this by using model-based reinforcement learning, where we approximate the transition function by using Bayesian inference. We base the posterior probability on observations of state transitions and a prior information about the environment, which in our case is the location of targets. This approach is different to the BA-POMDP model [Ross et al., 2007], where the model (of the transition and observation function) is represented by mixtures of Dirichlet distributions of transitions and observations count vectors, whereas our model (of only the transition function) is represented by Bayesian inference over multinomial distribution of transition counts and Dirichlet priors. We now formally define the posterior probability distribution which makes up the model of the transition function. In Table 6.1 we provide a list of used notations, which we then formally define.

θ	vector of parameters of multinomial distribution
ϕ	vector of transition counts (s, a, s')
ρ	vector of hyperparameters of Dirichlet prior, i.e., pseudo-observations
IS	information set of current states, i.e., states with non-zero beliefs

Table 6.1: List of notations used in BayesRQ.

We define a discrete random variable X as a function of states, then $P(X = s)$ is the probability of being in state s , noted as belief $b(s)$. This discrete random variable is defined over a given information set IS . The discrete probability distribution of X is parametrised by a vector θ , where $\sum_{i \in IS} \theta_i = 1$ and $P(X = s_i | \theta_i) = \theta_i$. The next step now is to compute the probability distribution of the random variable X given an information set IS , which in other words means obtaining beliefs $b(s)$ over the states in the information set IS . Thus, θ_i is the probability of being in state s_i . A key assumption in our model is that the defender can fully observe some of the state transitions represented by the transition tuple (s, a, s') , as a transition from a state s after taking an action a to a succeeding state s' . From these observed transitions the defender forms a vector of transition counts ϕ . The vector ϕ

is defined for each state-action pair as $\phi^{sa} = (\phi_1^{sa}, \dots, \phi_k^{sa})$, where ϕ_i^{sa} is the number of observed transitions from the state s after taking the action a to the next state $s'_i \in IS'$, and k is the size of the succeeding information set $|IS'|$.¹

Maximum likelihood estimate (MLE) We can now introduce the likelihood probability $P(\phi|\theta)$ as the probability of a transition vector ϕ given the parameter θ . The probability distribution over a finite number of states in a given information set IS is discrete and thus we assume a multinomial distribution with number of categories equal to the number of states i.e., the size of the given information set. We parametrise the multinomial distribution by parameter θ , we then write the probability mass function as:

$$P(\phi|\theta) \sim f(\phi|\theta) = \frac{n!}{\prod_{i \in IS} \phi_i!} \prod_{i \in IS} \theta_i^{\phi_i}, \quad (6.1)$$

where $n = \sum_{i \in IS} \phi_i$ and $\theta_i^{\phi_i}$ is θ_i raised to the power of ϕ_i . Note that $\frac{n!}{\prod_{i \in IS} \phi_i!}$ is the total number of possible observations of transition sequences given the transition vector ϕ . To find the value of parameter θ which makes the observed transitions *the most probable* we use the *maximum likelihood estimation* (MLE). Given the observed transitions ϕ we define MLE as:

$$\theta_{MLE} = \arg \max_{\theta} P(\phi|\theta) = \arg \max_{\theta} \prod_{i \in IS} f(\phi_i|\theta). \quad (6.2)$$

For a multinomial distribution this equation has a simple solution, where the most likely probability distribution is $\theta_{MLE} = (\frac{\phi_1}{n}, \dots, \frac{\phi_k}{n})$.

Maximum a posteriori (MAP) In spatial security games the defender might often have access to a priori information about the environment, which can help him in dealing with uncertainty. For example in the illegal rhino poaching problem such a priori knowledge can be the location of the rhinos (targets), which very likely influences the attacker behaviour. Therefore, it might be very useful to include such a priori information into forming the beliefs about states, which we do by using Bayesian inference. We encode the prior information as a priori probability which follows Dirichlet distribution $Dir(\rho)$, defined for hyperparameters ρ . $Dir(\rho)$ is a probability distribution over parameters θ of a multinomial distribution and is also its conjugate prior. The vector of hyperparameters ρ can be seen as pseudo-

¹For better readability we use ϕ_i to denote $\phi_{s'_i}^{sa}$ for a given state s , an action a and a succeeding state s' .

observations to complement the actual observed transitions i.e., the transition counts ϕ . $Dir(\rho)$ is defined using the Γ function as:

$$Dir(\theta|\rho) = \frac{\Gamma(\sum_i^k \rho_i)}{\prod_i^k \Gamma(\rho_i)} \prod_{i=1}^k \theta_i^{\rho_i-1}. \quad (6.3)$$

In the previous paragraph we described the likelihood, modelled as a multinomial distribution, using the observed transitions (data) ϕ (see Equation 6.1). We can thus define the posterior using Bayes rule as:

$$P(\theta|\phi) = Dir(\theta|\phi) \propto Multi(\phi|\theta) Dir(\rho) \quad (6.4)$$

We can then write $P(\theta|\phi) = Dir(\theta|\phi + \rho)$, because Dirichlet distribution is a conjugate to multinomial distribution.

Our main goal is to derive beliefs about states, for which we do not use the full posterior distribution due to intractability and high computation costs, but instead, we focus on a point estimate of the posterior. The state belief is then defined as the maximal posterior probability, known as the *maximum a posteriori* (MAP).² MAP is computed based on the data (observations of the state transitions) and the prior information (pseudo-observations); where with the increasing number of the real observations of the transitions, the effect of a priori distribution on a posteriori distribution diminishes. MAP is defined for a Dirichlet posterior as:

$$\theta_{MAP} = \arg \max_{\theta} P(\theta|\phi) \rightarrow \theta_i^{MAP} = \frac{\phi_i + \rho_i - 1}{n + \sum_{j \in IS} (\rho_j - 1)}. \quad (6.5)$$

Depending on whether we have an access to a prior information we use MLE or MAP to derive the model of the transition function and then to obtain the beliefs about states. We assume that the transitions are deterministic but unknown, e.g., in the spatial security domain if the agents decide to move to a particular location, they move there with probability 1. Such a move is fully observed on the defender side but uncertain on the attacker side since the defender does not know the attacker's behaviour. Therefore, Bayesian inference is used to learn the attacker's behaviour and as a result the transition function. We denote the belief state $b^{oa}(s)$, which is the probability of being in the state s given the observation o and

²Note that when deriving point estimates of posterior distribution we do not need marginal distribution of the data (the normalising constant) $P(\phi)$.

action a . The observation o consists of the previous information set and the state transition counts ϕ . Hence, we do not need to explicitly learn the observation function $O(o|s', a)$. We can now obtain the belief $b^{oa}(s')$ of the succeeding state s' given an observation o and an action a . The observation defines the belief $b(s)$ about the current state s , transition counts $\phi_{s'}^{sa}$ and priors ρ as:

$$b^{oa}(s') = \sum_{i \in IS} b(s_i) \theta_{s'}^{MAP} = \sum_{i \in IS} b(s_i) \frac{\phi_{s'} + \rho_{s'} - 1}{n_{s_i, a, \cdot} + \sum_{j \in IS'} (\rho_j - 1)}, \quad (6.6)$$

where $n_{s, a, \cdot}$ is the sum of the transition counts over all the succeeding states from a state s after taking an action a . This represents reasoning between two information sets, the current IS and the succeeding IS' , see for example Figure 6.2.

Designing hyperparameters of prior distribution We now discuss the setting of the hyperparameters ρ . In general this can be any a priori information we might have about the environment (domain knowledge), which might influence the attacker behaviour. In spatial security games we assume that the attacker behaviour is steered by the location of the targets which is known information to both players at the beginning of the game. Therefore, we define the prior for each node (location) as $\rho_{node} = \frac{1}{SP(node, target) + 1} * prior_confidence$, where $SP(node, target)$ is the shortest path to the nearest target from the given node, and $prior_confidence$ is a parameter describing how much weight we assign to the prior and thus determines the confidence of such prior information in comparison to the actual observations. The value of $prior_confidence$ can be thought of as the number of pseudo-observations we make before the game starts. Note that the prior is defined for a location of the attacker ignoring the location of the defender. This simplification comes from the assumption that the attacker cannot fully observe the defender location in a given game episode (but knows the past moves) and is mainly steered by the location of the targets.

Saving transition counts in partial observability In every time step the defender updates the state-action transition counts $\phi_{s'}^{sa}$ for the respective transition tuple (s, a, s') . In the case when he cannot fully observe the current or/and the succeeding state he updates the transition counts $\phi_{s'}^{sa}$ proportionally to the beliefs as $\phi_{s'}^{sa} += b(s)b(s')$. Therefore, the stronger the belief about the particular states is the more he updates the corresponding value in the vector ϕ . Note that for fully observed states s and s' the update is equal to 1.

6.3 Q-learning with Bayesian Inference: BayesRQ

The main goal in spatial security games is to find an effective policy to deal with the partially observable attacker. We obtain the defender policy by using temporal difference reinforcement learning method. As we explain in Section 2.3 there are several ways to apply TD learning in multi-agent environments with partially observable agents. We can either totally ignore the other agents and model them as part of the environment or explicitly model them in the state-action value function. To keep the Markov property we define the environment state as the location of the defender and the attacker. However, in the experiments we also compare the proposed method with the case where the attacker is completely ignored and define the state as only the location of the defender, which obviously breaks the Markov property.

Our approach is based on a version of independent learning, where we cannot observe the actions of the attacker and thus cannot learn in joint-action manner. We assume deterministic transitions, i.e., the players always move to the desired location. We model the attacker as a part of the transition function, which we learn during the interaction. In the previous section we have proposed a way to learn a model of the attacker behaviour by using Bayesian inference over (partially) observed state-action transitions and potentially a priori information. We base our approach on the idea of Replicated Q-learning (RQ) proposed by Littman et al. [1995], which modifies the standard temporal difference learning method Q-learning for belief states. In Replicated Q-learning the state-action value function update has the form

$$Q(s, a) = (1 - b(s)\alpha)Q(s, a) + b(s)\alpha(r + \gamma V(s')), \quad (6.7)$$

where each state-action pair is updated proportionally to the belief about that state. In the case of having a sufficient number of observations or being able to observe the underlying state, the RQ method reduces to standard Q-learning, which under certain conditions converges to the optimal Q-function [Tsitsiklis, 1994; Jaakkola et al., 1994]. We combine the RQ algorithm with Bayesian inference for deriving the beliefs about states, presenting a novel method, which we call BayesRQ. BayesRQ is an algorithm suitable for the partially observable spatial security games, where the defender can sometimes fully observe the attacker. We make use of the assumed occasional observability in spatial security games, which enables us to sometimes observe the underlying system state, i.e., the location of both

Algorithm 1 BayesRQ

-
- 1: **Input:** priors ρ , parameters α, γ
 - 2: **Init:** $s_0, Q(s, a) = 0, \phi_{s'}^{sa} = 0 \quad \forall s, s' \in S, \forall a \in A, b(s)$ uniform over IS
 - 3: **Def:** state s is defined as position of both defender and attacker
 - 4: **for** t in episode **do**
 - 5: ϵ -greedy $\rightarrow \epsilon$: random action and $(1 - \epsilon)$: $a = \arg \max_a \sum_{i \in IS} b(s_i) Q(s_i, a)$
 - 6: $\forall s$: $Q(s, a) = (1 - b(s)\alpha)Q(s, a) + b(s)\alpha(r + \gamma V(s'))$
 - 7: where $V(s') = \sum_{j \in IS'} b(s'_j) \max_a Q(s'_j, a)$
 - 8: $b^{oa}(s') = \sum_{i \in IS} b(s_i) \frac{\phi_{s'}^{sa} + \rho_{s'} - 1}{n + \sum_{j \in IS'} (\rho_j - 1)}$
 - 9: $\forall s, s'$: $\phi_{s'}^{sa} += b(s)b(s')$
-

players. Since the location of the defender is always known, observing the underlying state is possible only when observing the attacker location. We present BayesRQ in Algorithm 1. The action-selection on line 4 is an ϵ -greedy strategy; with probability ϵ we choose a random action and with probability $1 - \epsilon$ we choose an action which maximises over all possible actions given the current information set IS of states with non-zero beliefs. Thus, an action a from a state s , is more likely to be chosen with increasing probability of being in that state s and increasing corresponding Q-value for that state and action. On line 5 we update Q-values for all the states and chosen action using the beliefs about the states (non-zero beliefs only over the states in the current information set). The learning rate α is multiplied by the belief we have about the state; the lower the probability of being in a particular state the less we update the Q-value for that state and a chosen action and vice-versa (greater update for greater belief). The value function of the next state on line 6 is a sum over maximal Q-values of the succeeding states weighted by the probability (belief) of transitioning to those states; all the possible succeeding states are defined by the succeeding information set IS' . The belief update on line 7 uses the maximum of the posterior probability distribution (MAP) as explained in Equation 6.6. Finally, on line 8 we update the transition count vector ϕ for all states in the current information set, i.e., $\forall s \in IS$ and all succeeding states in the succeeding information set, i.e., $\forall s' \in IS'$ and for the chosen defender action a . Note that when we can fully observe the attacker, i.e., the information sets are singletons, the proposed algorithm BayesRQ becomes standard Q-learning.

6.4 Experiments with the BayesRQ Algorithm

In this section we experimentally evaluate the proposed algorithm BayesRQ. We compare several cases of observability of the attacker, from the extreme case where the attacker is never observed, to observing him every 2, 3 or 4 steps, to observing him in every step. We also experiment with random observability by defining a probability of observing the attacker per step. For the case when the attacker can never be observed, the algorithm BayesRQ cannot be used due to too large information sets and thus slow performance. Instead, we use the standard Q-learning, where we define the state as only the location of the defender, i.e., ignoring the attacker. Obviously such approach breaks the Markov property and thus we cannot relate the BayesRQ algorithm to any of the well-known convergence proofs [Tsitsiklis, 1994; Jaakkola et al., 1994], however in some settings such approach, which is a variant of *independent learning* (see Section 2.3), can still perform very well.

We consider a grid world which can for example represent a wildlife reservation with some targets (e.g., rhinos) which the attacker wants to attack. In Figure 6.3 we show a grid world, where the defender starts at the left top corner and the attacker starts at the right bottom corner. The black tiles with white stripes represent some obstacles and the tiles with white crosses represent the targets. The attacker’s goal is to get to the targets (successful attack) and the defender’s goal is to apprehend the attacker, i.e., to be in the same tile as the attacker. The actions of the players are *up*, *down*, *left*, *right*. The heatmap shows the defender visits in each tile, the warmer the colour the more frequently the defender went to that tile. The black dots show the attacker visits in each tile, the bigger the dot the more frequently the attacker went to that tile. Each game ends when either the defender apprehends the attacker or the attacker successfully attacks the target. In Figure 6.3 we show the map for four different cases of observability of the attacker. From full observability (left top), where BayesRQ becomes the standard Q-learning, to BayesRQ with observing the attacker every 3 time steps (right top), every 4 time steps (left bottom). Finally in right bottom panel of Figure 6.3 we assume no observability of the attacker, using Q-learning while ignoring the attacker location, i.e., defining the state as only the defender location. We can see that the attacker needs to alter between those two targets and cannot only focus on one target no matter what observability capabilities the defender has in order not to get exploited. One can notice that for low observability of the attacker, the attacker gets more often to the more distant target at location $[4, 0]$ before being apprehended at location

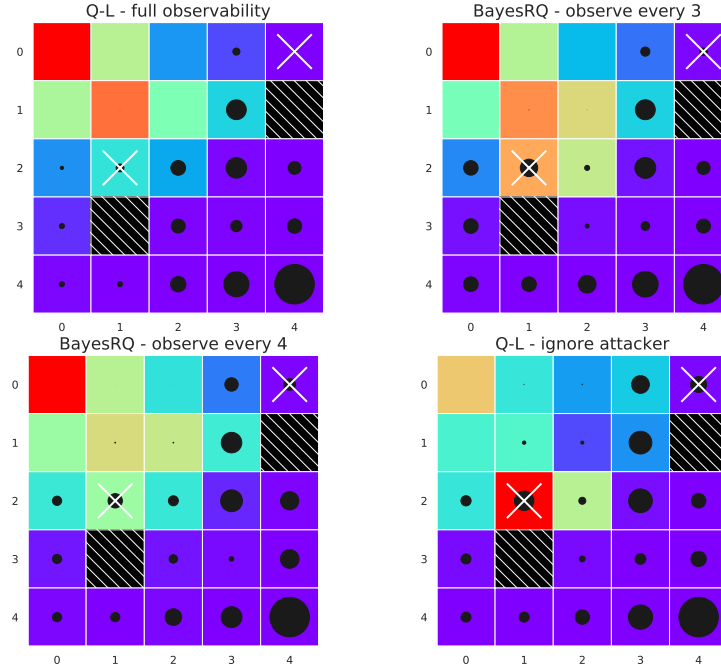


Figure 6.3: Grid World: The defender starts in left top corner and aims to apprehend the attacker, the attacker starts at right bottom corner and aims to attack the targets (white crosses) avoiding the obstacles (black tiles with white diagonal stripes). Attacker attacks more the right top (more distant) target with the defender’s decreasing observability of the attacker. The heatmap shows the number of visits to each state (red to purple, purple is none), the black dots show attacker visits in each tile (the bigger the dot the higher number of visits).

[3, 1]; compare the size of the dots in location [3, 1] and [3, 0]. Expectedly, with decreasing observability of the attacker, the attacker more successfully attacks both of the targets, i.e., bigger black dots in the tiles with white crosses. Note that in the case of ignoring the attacker (no observability) the preferred strategy for the defender is to move around the closer target. We can see the performance of these experiments in terms of the defender wins in the following subsection in Figure 6.4.

Performance We compare the performance for different levels of observability of the attacker on the Grid World example from Figure 6.3. We assume a finite horizon game and set the discount factor $\gamma = 1$. For all the methods we use a classic setting for reinforcement learning algorithms, learning rate $\alpha = 0.05$ and exploration rate $\epsilon = 0.1$. We consider a training time of 1,000 episodes, where the shown results are averaged over 1,000 trials

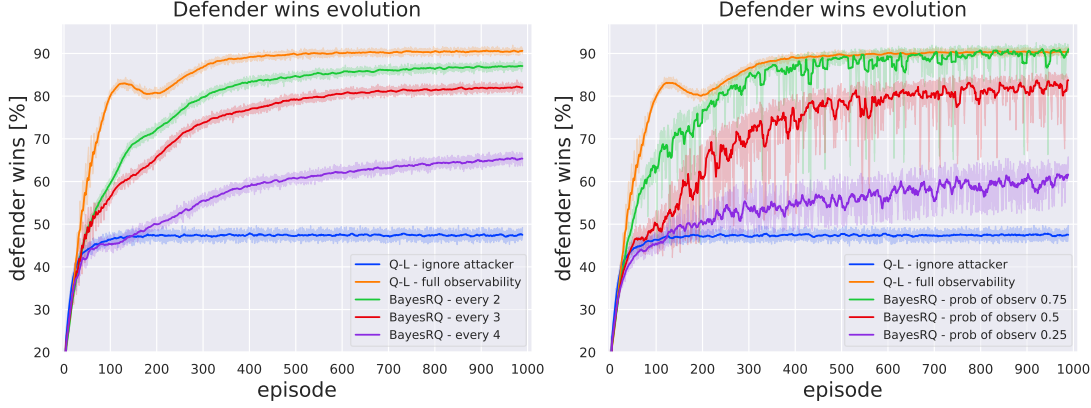


Figure 6.4: Observability analysis: Defender wins ratio for BayesRQ with limited observability of the attacker. Left figure: observing attacker periodically every $[0, 1, 2, 3, 4]$ time steps, right figure: probability of observing the attacker is $[0, 0.25, 0.5, 0.75, 1]$ in each state.

for which we show 95% confidence intervals. In Figure 6.4 we show the performance in terms of the defender wins, i.e., the percentage of games where the defender apprehended the attacker. We show five modes of observability in each figure, in the left figure of Figure 6.4: (i) no observability of the attacker, where we use the standard Q-learning with state definition of only the defender's location, i.e., ignoring the attacker, (ii) full observability, i.e., observing the attacker location in every time step and thus knowing the underlying state consisting of the location of the defender and the attacker, for which BayesRQ becomes Q-learning, (iii) BayesRQ with observing the attacker periodically every 2 time steps, (iv) BayesRQ with observing the attacker periodically every 3 time steps and (v) BayesRQ with observing the attacker periodically every 4 time steps. We can see that our algorithm BayesRQ can very well deal with the occasional partial observability of the attacker location. Especially for the case of observing the attacker every 2 or 3 time steps, the performance of BayesRQ is very close to the full observability case. In the right panel of Figure 6.4 we show an experiment with a different type of observability, where we no longer assume periodical observability but we assign a probability of observing the attacker in each state. The two extreme cases are the same as in the previous figure, i.e., full observability and no observability, but now we plot the performance for BayesRQ for cases where we get to observe the attacker with probability 0.75, 0.5 or 0.25 in each state. One can compare the left and right figures in Figure 6.4, where for example observing the attacker every 4 time steps is comparable with the case of observing the attacker with probability 0.25 in each

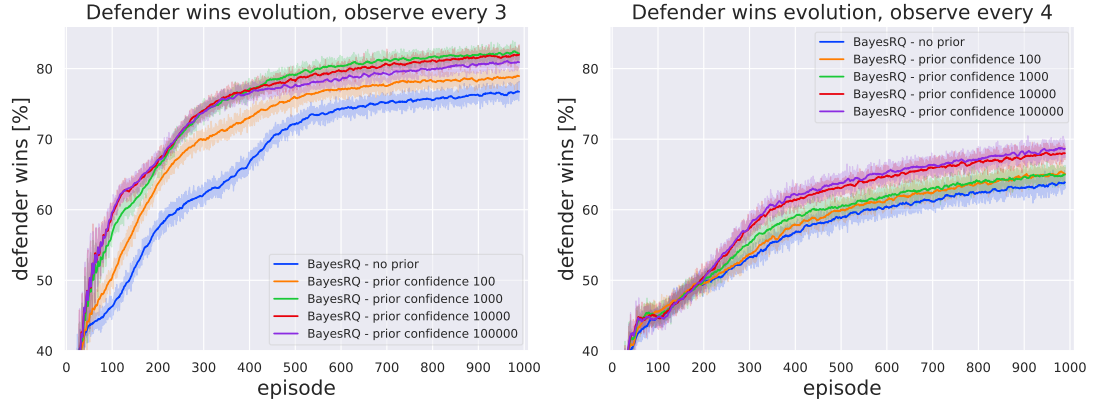


Figure 6.5: Priors analysis: Defender ratio of wins for BayesRQ, observing the attacker every 3 steps.

state (purple curves). The experiment with the random observability further demonstrates the efficient use of the partial information in the BayesRQ algorithm. We can see that the performance of BayesRQ in the random observability case has a higher variance but is still superior to the case where we ignore the attacker, which is expected. By comparing the full observability case with the partial observability cases we can conclude that BayesRQ efficiently makes use of the gathered information and can well reason about the unobserved states. Especially for the rather frequent observations of the attacker (observing every 2 or 3 time steps or probability of observing the attacker 0.75 or 0.5) BayesRQ approaches the performance of the full observability case.

Arguably, periodical observability of the attacker might cause synchronisation in moves between the defender and the attacker and the performance of the algorithm might depend on the chosen map. Therefore, we also experimented with the random observability, which confirms the strong performance of BayesRQ.

Prior distribution analysis As discussed previously, BayesRQ uses a priori information about the attacker location, in our case this information is derived from the knowledge about the location of the targets, where we assume an incentive of the attacker to move closer to those targets. Therefore, the setting of such prior is potentially important. In our model we introduce the parameter of prior confidence as described in Section 6.2.2. In Figure 6.5 we test the influence of different settings of this parameter on overall performance. We show the case where we do not assume any prior knowledge (prior confidence equal to 0)

in which case we use the maximum likelihood estimate (MLE). And then we show different settings of the prior confidence (100; 1,000; 10,000; 100,000), i.e., using the maximum a posteriori (MAP). In the left figure of Figure 6.5 we assume observing the attacker every 3 time steps and in the right figure we get to observe the attacker every 4 time steps. We can see that the setting of the prior confidence has a significant influence on the performance of BayesRQ. Using no prior information leads to the worst performance, confirming that the prior is meaningful in this case. An interesting observation is that in the case of using no prior or low prior confidence (100) the performance stays inferior even in the later episodes, the reason for that might be a form of exploitation of the defender by the attacker, where the attacker learns faster about the best paths to the targets. We can also see that the effect of different settings of prior confidence changes with observing the attacker only every 4th time step (right figure), where we can observe a better performance with even higher prior confidence (10,000 and 100,000). The reason might be that the defender has a higher uncertainty about the attacker and thus the effect of the very limited observations is diminished and he benefits more from a stronger prior knowledge. In our experiments in Figure 6.4 we set the prior confidence to 1,000. We conclude that the setting of the prior distribution is important and should be carefully tuned when designing the algorithm for a given domain.

6.5 Discussion

In this chapter we have proposed a novel approach to learning effective strategies in partially observable spatial security domains using a priori knowledge and occasional observations of the attacker location. This novel method learns a model of the attacker behaviour and effectively uses it in a temporal difference style learning algorithm. We combine Bayesian inference with the Replicated Q-learning method and propose BayesRQ, which learns an effective defender strategy to significantly improve the chance of apprehending an adversarial attacker and thus mitigating potential attacks on targets.

Our approach is based on the assumption of occasional observation of the attacker location, where especially in case of frequent observations of the attacker location our method comes close to performance of a full observability case. This assumption is motivated by many spatial security domains, where the attacker location can for example be obtained from surveillance. The proposed solution can for example be applied to the illegal rhino poaching problem where rangers aim to apprehend poachers and thus minimise the number

of poached rhinos, preventing potential future extinction of the species. In the illegal rhino poaching domain the frequent observability assumptions are common, where the attacker location can be for example obtained from the people living in the area or from drone surveillance [Montesh, 2013]. Analogously, our approach can be applied to other spatial security domains with similar observability properties.

Our experimental evaluation shows that BayesRQ can deal effectively with partial observability. Especially, our solution is able to successfully bridge the occasional inability of observing the attacker location, where we can closely approach the performance of the full observability case and learn an effective defender strategy. We experimented with two different observability modes: (i) occasional full observability of the attacker’s location in the form of periodical observability, i.e., every fixed number of time steps and (ii) in the form of random observability, i.e., with some probability we get to observe the attacker’s location in every time step. Experiments show strong performance of BayesRQ on the proposed grid world for different levels of observability. Moreover, we experimented with different settings of prior distribution, which influences the learning process. We showed that our chosen prior distribution is meaningful for our grid world domain and helps to learn effective strategies. Note that the prior distribution is domain dependent and can be designed from different a priori knowledge. The proposed method combines Bayesian inference with the Q-learning algorithm, but the general idea can be applied to other temporal difference learning methods such as SARSA, Expected SARSA and other reinforcement learning methods.

Our approach is effective when the observation of the attacker is rather frequent. For more rare observations the size of the information sets increases significantly, causing the algorithm to slow down. Therefore, we point out that the application area should possess the property of frequent observations of the attacker for effective use of our approach.

7

Conclusion

In this final chapter we conclude the thesis by summarising the main contributions and answering the research questions posed in Chapter 1. We then propose future directions for further research, with reflect on the limitations of this work.

7.1 Contributions and Answers to the Research Questions

In this section we answer the 3 research questions we defined in Section 1.2, where we also posed the problem statement. We now give the answers and refer to the respective chapters of this thesis.

Question 1.A: *How can we model a complex environment such as the space debris removal problem from a game-theoretic and learning perspective in order to understand how agents can optimise their behaviour?*

An important first step when modelling complex environments from a game-theoretic perspective is the evaluation of the impact of different strategies taken by the agents; what

effects different actions have on the environment and on the other agents. One powerful way to evaluate that, studied by *empirical game theoretical analysis* [Wellman, 2006; Tuyls et al., 2018], is in many domains to develop a simulator of such a complex environment, especially if the environment allows for wide range of future evolutions. This is the case in the space debris removal problem, investigated in Chapter 4, where we implemented a high-fidelity space debris simulator, which we fully describe in Appendix A. While the simulator aims to be realistic, it is also computationally demanding and thus not possible to be effectively used for evaluation of different strategies emerging from various models of multi-agent interaction. Therefore, we developed a surrogate model based on the high-fidelity simulator, which enables us to evaluate complex strategies of potentially multiple agents. We then presented an effective way to approach the modelling of such a complex environment by considering multiple models of agent interaction such as single- and multi-agent models and single-state (stage) and multi-state models. These models represent various realistic scenarios such as full cooperation among the agents or purely self-interested behaviour. All these models produce different types of strategies, such as Nash equilibria or optimal strategies in terms of maximal return. Furthermore for the dynamic multi-state model we showed how agents can learn effective behaviour from interactions with the environment using reinforcement learning techniques. Therefore, such an analysis brings deeper insights to the complex dynamics of the agent interactions with the environment and with each other.

Question 1.B: *How can we compare various solutions that emerge when following different modelling choices of the agent interaction mechanism and evaluate their effectiveness in complex domains?*

In Chapter 4 we proposed an evaluation methodology to compare the quality of solutions from following different modelling choices, which improves the understanding of the dynamics of strategy formation among multiple players in the space debris removal problem. We compared single- and multi-agents models, assuming either static one-shot strategies or dynamic strategies, where agents dynamically decide on actions over many time steps. Moreover, we analysed various solution types obtained from following different methods such as exactly computed solution in the form of Nash equilibria or learned solution obtained by applying the Q-learning method. Especially in domains where sustainability of the environment is crucial, important evaluation metrics to consider are social welfare or

fairness. Fairness expresses the proportionality between the costs of invested efforts and gains from them. Furthermore, we focused on *price of anarchy* (PoA) analysis, measuring the ineffectiveness of selfish behaviour compared to cooperative behaviour, which are both realistic scenarios in many sustainability domains.

In the space debris removal problem we showed how a selfish behaviour can be quite costly compared to centralised cooperative behaviour in terms of social welfare. Although, selfish rational behaviour is more likely to appear in the space debris removal problem, the space actors should try to reach a consensus and act cooperatively, which would decrease the costs and would be more sustainable due to increased stability of the centralised behaviour. Static strategy solutions in the space debris removal problem are less effective due to smaller flexibility compared to the dynamic ones, however static strategies are simpler to obtain and arguably to abide and thus might be preferable in certain situations.

Question 2.A: *How can we design a robust learning process against random or adversarial perturbations in critical systems with risky states?*

In Chapter 5 we proposed a novel κ operator which embeds robustness into temporal difference learning algorithms for both single- and multi-agent environments. The robust behaviour is then attained by modifying the target of temporal difference learning methods. We base the robust operator on a priori known, or estimated information about an external control over the system or its parts which represents the potential severe attacks or critical failures. In critical domains with risky states there is often the threat of individual parts of the network being compromised which might cause severe impact on the whole system. We showed how robust strategies can be learned by interacting with the environment even before observing an attack or a failure.

Question 2.B: *To what extent can we guarantee any convergence to an optimal solution when learning policies, which are robust against perturbations in domains with risky states?*

In Chapter 5 we presented convergence proofs for the proposed robust temporal difference learning algorithms using our operator κ . We proved convergence of the methods to the optimal Q-value function of the original Markov decision process and also convergence to the robust Q-value function induced by the operator κ of the generalised Markov decision process. We showed under which conditions the single- and multi-agent methods converge

to those two fixed points. Therefore, we can guarantee stability of the proposed robust learning process, which increases the applicability of our approach.

Question 3.A: *How can we model complex adversarial agents in spatial security domains in order to mitigate the threats they pose?*

A powerful approach to finding effective behaviour in partially observable environments is model-based reinforcement learning. In Chapter 6 we analysed the illegal rhino poaching problem as an instance of spatial security domains. Threats in such domains can be mitigated by effective behaviour of the controlled agent or group of agents (the defender), who aim to apprehend adversarial attackers. We assume that the system state is represented by the locations of the defender and the attacker; therefore we can model the attacker behaviour as part of the transition function, which we learn. Learning the transition function is based on potentially known a priori information about the attacker behaviour and on occasional observations of the attacker location. We proposed an effective approach based on temporal difference learning and Bayesian inference, which can learn a model of the attacker in order to derive effective strategies to apprehend the attacker.

Question 3.B: *To what extent can we make use of an occasional full observation of the adversarial attacker in the spatial security domains?*

In many spatial security domains such as illegal rhino poaching the defender gets to occasionally observe the attacker location, for example from drone surveillance [Montesh, 2013]. Such partial knowledge can be used to form effective defender strategies. In Chapter 6 we proposed a novel approach which can make use of such knowledge in order to learn powerful defender strategies, which can mitigate the threat of harmful attacks. We derived a new model-based reinforcement learning approach BayesRQ, which makes use of a priori information about the attacker behaviour and of occasional observations of the attacker. Especially for the case of frequent attacker observations, our method can bridge the occasional inability to observe the attacker location and can closely approximate the performance of the model with full observability of the attacker. Our method opens a new way to learning effective defender behaviour in spatial security domains where the attacker can be occasionally observed.

7.2 Limitations and Perspectives for Future Research

In this thesis we have examined applicability of the multi-agent learning paradigm to real-world domains. The focus of this thesis was on some of the most pressing issues of real-world applications of multi-agent learning methods such as the challenge of modelling complex environments and agent interactions, designing robust learning processes or modelling adversarial agents, especially in domains where security and sustainability is paramount. Nonetheless, there still exist other obstacles for multi-agent learning methods to be fully applicable to real-world domains. We now discuss several limitations and propose future directions of the work presented in this thesis.

Modelling and Learning in the Space Debris Removal Problem

In Chapter 4 we have proposed a methodology to model a complex environment and design and compare several models of multi-agent interactions. We focused on the complex domain of the space debris removal problem, which we modelled from a game-theoretic perspective, presenting new insights into the complex dynamics of multi-agent decision making. The input data for our single- and multi-agent and static and dynamic models came from our high-fidelity debris simulator. However, projecting the future evolution of space debris itself is a very complex problem with many unknown variables and inputs, and therefore some necessary simplifications and assumptions have been made. Despite these simplifications the simulation is computationally demanding, which makes it difficult to obtain the necessary number of Monte Carlo runs, especially for larger games. Therefore, by using our surrogate model built on top of the space debris simulator, we were able to compare several complex models of agent interaction. Even though we assumed the surrogate model to be a close approximation of the simulator, some level of abstraction was introduced which necessarily lead to a loss of some information value. In future work one could investigate a broader range of scenarios (e.g., launch parameters) for the space debris simulator. The surrogate model is constructed and analysed for the *conservative* scenario, described in Appendix A.4. However, as discussed in that section, several scenarios can be envisioned that each will lead to a different projected evolution of the space debris environment. Using the high-fidelity simulator, it is conceptually easy (but computationally demanding) to construct new surrogate models for these different scenarios. When computational power is available though, the methodology we developed will make it easy to run the required Monte Carlo

simulations to build a new model, which can then be analysed in the same fashion as we have done in this thesis for the conservative scenario.

Another extension would concern the addition of mega constellations to the simulator, which we discuss in Appendix A.4.6, in addition to the four classes of satellites we have considered (see Appendix A.4). Mega constellations are currently being considered as a new addition to traditional satellite operation. Besides adding significant numbers of satellites to the space environment, mega constellations would constantly replenish their supply of satellites over a long period of time. If left unmitigated, this has been shown to have a profound effect on the space environment [Rossi et al., 2017] and is thus worth including in the simulation.

Robust Learning in Critical Systems with Risky States

In Chapter 5 we have presented a novel approach to learning robust and effective strategies in critical domains, both from a single- and multi-agent perspective. This line of work opens up a new path of incorporating robustness into reinforcement learning methods leading to safer policies. We have presented a new robust operator κ , which can be combined with many different methods. However, in this thesis the operator is demonstrated on a classic on- and off-policy one step temporal difference learning methods, which is a first step in showing its usability. For the multi-agent setting we only considered full communication among cooperating agents, furthermore, as we briefly discussed, one could also consider a more complex setting where agents cannot communicate and instead need to best-respond to policy estimation of other agents. In future work we suggest to start from a simple policy estimation like fictitious play, but more complex policy estimators could be assumed to better model the complicated multi-agent interaction.

We have proposed a simple model of control transition defining the potential significant rare events, however there are several interesting directions for future work, extending the control transition function. One possibility would be extending the control space, allowing for more agents being attacked or malfunctioning with different intensity. We defined the probability of control depending on the state, but more parameters could be introduced, e.g., making the control to depend on time. A state-dependent or time-dependent control is motivated by the fact that in practice some states might be critical and more prone to malicious attacks or malfunction in different time steps. Moreover, in future work the operator κ could be applied to other state-of-the-art methods like $\text{Retrace}(\lambda)$ [Munos et al.,

2016] to consider multi-step updates, $Q(\sigma)$ [De Asis et al., 2018] to assume mixed updates or combining the operator κ with the options framework [Sutton et al., 1999; Bacon et al., 2017]. We also hinted a generalised model of the control transition, which would allow for modelling of complex dynamics of control transition. We have only briefly touched upon this advanced model but deeper analysis could bring further insights into more complex scenarios of control, where one could assume several Q-value functions formed by different policies of multiple agents.

Finally, throughout this thesis we focused on discrete state and action spaces to model an environment, which is a first important step to underpin the theory, however in many complex real-world applications such assumption is limiting and would need to be extended to continuous state and action spaces. For such cases one could extend our approach by using a non-linear function approximation to represent the value functions of continuous state and actions spaces. Therefore, in order to increase the applicability of our methods, one could extend the methods with neural network types of learning, e.g., using DQN [Mnih et al., 2013]. Such extensions would further narrow the reality gap and would allow for learning more complex policies, where we believe our approach could prove even more competitive.

Learning When Facing Adversarial Agents in Spatial Security Domains

In Chapter 6 we have proposed a novel way of learning effective strategies in spatial security domains by learning a model of the attacker. Our approach is specially tailored to domains where the attacker location can be frequently observed. Although, our work has clear limitations in requirement of frequent observations of the attacker, we claim that such conditions appear in some spatial security domains and thus are interesting to be studied. The proposed method BayesRQ is based on a classic Q-learning method, however the presented approach of combining Bayesian inference with a reinforcement learning method could be extended to other learning methods. We described learning of a model of the attacker by using Bayesian inference. Such a model is then used to learn the state-action value function by using a temporal difference learning method. We update the state-action values proportionally to the beliefs, however in future work we could update the value function directly for belief states and thus learn a continuous belief state-action value function.


Our approach is based on Bayesian inference, thus one can define a priori distribution,


which then influences the learning process. We showed how such distribution can be designed in order to be meaningful. However, a priori distribution needs to be chosen in a domain specific manner and fine tuned by gathering a priori knowledge about the model. There is a wide range of opportunities for designing more complex a priori distribution and analysing its effects on the overall learning process, which would be an interesting further research direction. We proposed an example of a prior knowledge about the environment motivated by the domain of illegal rhino poaching but this could be extended to encode more complex a priori information.

As mentioned throughout this thesis obtaining any theoretical guarantees in multi-agent learning is a challenging problem. Nevertheless, one could aim to prove for example convergence properties of the proposed algorithm BayesRQ based on the well-known proofs of single-agent classic temporal difference learning methods (e.g., Tsitsiklis [1994]; Jaakkola et al. [1994]). However, such an extension to partially observable domains is non-trivial.

For the studied target domain of illegal rhino poaching it is suitable to use a discrete state space, however a more complex approach could be considered, where the state space would be continuous. Such extension would allow modelling of more complex environments, where the state cannot be accurately defined in discrete manner. Similarly, we only considered discrete action space for the players, where more granular approach might be beneficial for certain environments. Value functions of continuous action and state spaces could be modelled by a non-linear function approximation such as a neural network. This approach might be also necessary for larger domains, speeding up the methods and thus further increasing their usability for real-world applications.

List of Figures

1.1	Problem statement structure.	7
2.1	Relation between models of agent(s) interaction with an environment. In green are multi-agent models with a single state (stage), in yellow are single-agent models and in red is their generalisation: multi-agent model with multiple states.	14
2.2	Markov decision process: The agent interaction with an environment. Taken from Sutton and Barto [1998].	24
4.1	Projected evolution of the (a) total number of objects and (b) cumulative lost assets, assuming the complex launch model, in the next 100 years for different removal strategies, e.g. <i>above 8000</i> - removing all the objects causing in expectation a collision producing more than 8000 debris pieces.	53
5.1	The relationship between the learning targets of different algorithms in the limits of their parameters. On-policy methods are in green, off-policy methods in orange.	90
5.2	Cliff Walking: The agent needs to get from the start [S] to the goal [G], avoiding the cliff (grey tiles).	96
5.3	The Puddle World: $Q(\kappa)$ learns a safer path with increasing \varkappa . Puddles are dark blue, the arrows show the optimal actions on the learned path, and the heatmap shows the number of visits to each state ( , blue is none).	97

5.4	Cliff Walking (single-agent) in first row and Puddle World (multi-agent) in second row. Deterministic environment (first column), 10 % stochastic environment (second column) and 10 % attack while training (third column). ϵ -greedy policy with fixed $\epsilon = 0.1$. Early performance - dashed lines (100 episodes), converged performance - solid lines (100,000 episodes).	98
5.5	Varying probability of attack: Cliff Walking (left), Puddle World (right), trained on 100k episodes, tested on 50k episodes, $\alpha = 0.1$, $\epsilon = 0.1$	100
5.6	Robustness analysis: Cliff Walking (left), Puddle World (right), trained on 100k episodes, tested on 50k episodes, $\alpha = 0.1$, $\epsilon = 0.1$, $\kappa = 0.1$	101
6.1	Example of modelling a wildlife reservation (Kruger park in South Africa) as a grid world. [www.safari.com/kruger-national-park/maps/kruger-park-far-south-section].	109
6.2	Example of states in information sets for the case where neither the current state nor the succeeding state is fully observed. We need to reason over two information sets; the current one IS and the succeeding one IS' . The defender is in location 1 and chooses the action <i>down</i> (D), the attacker is either in location 5 or 7.	111
6.3	Grid World: The defender starts in left top corner and aims to apprehend the attacker, the attacker starts at right bottom corner and aims to attack the targets (white crosses) avoiding the obstacles (black tiles with white diagonal stripes). Attacker attacks more the right top (more distant) target with the defender's decreasing observability of the attacker. The heatmap shows the number of visits to each state ( , purple is none), the black dots show attacker visits in each tile (the bigger the dot the higher number of visits).	119
6.4	Observability analysis: Defender wins ratio for BayesRQ with limited observability of the attacker. Left figure: observing attacker periodically every $[0, 1, 2, 3, 4]$ time steps, right figure: probability of observing the attacker is $[0, 0.25, 0.5, 0.75, 1]$ in each state.	120
6.5	Priors analysis: Defender ratio of wins for BayesRQ, observing the attacker every 3 steps.	121

A.1	Debris clouds from two sample collisions. Each dot is one piece of debris; the size represents their mass. Left figures show the clouds directly after the collision and right figures after 10 years. We can see how the objects in low altitudes decay.	161
A.2	Orbital decay of a sample debris object in Earth's atmosphere around 6371 km (Earth radius) caused by atmospheric drag.	162
A.3	Orbital inclination distribution of objects being launched in last 10 years. .	162
A.4	Distribution of osculating elements (normalised to 1) of all important assets (red) and debris (blue) in LEO which are currently less than 10 years old. Important assets are active satellites. We can observe the most common inclination and eccentricity orbital parameters.	163
A.5	Comparing spatial density predictions in LEO with related work of Liou and Johnson [2009]. Our model is in line with the related work.	164
A.6	<i>Mass to orbit</i> defining the total mass launched per year into low Earth orbit in relation to baseline mass launched in year 2000 (M_{2000}) for different values of α . For example the green curve represents a doubling of the annual mass to orbit over the next 100 years.	166
A.7	Distributions of semi-major axis, inclination, and eccentricity of objects filtered from current space catalogue from last 20 years, from which we sample new orbital elements for objects to be launched. The remaining orbital elements are sampled uniformly at random from a given range. . . .	169
A.8	Market share and mass to orbit function for the conservative, moderate and aggressive launch scenarios as specified in Table A.1. The conservative scenario assumes "business as usual" with constant launch mass and slower technical progress, however the aggressive scenario assumes a fast technological development with emphasis on miniaturisation of spacecrafts. . . .	174
B.1	Prediction of the development of the number of important assets (active satellites) for each player based on the assumption of a 0.5% yearly growth in future launches.	177
B.2	Debris evolution for next 150 years considering different strategies. The y-axis depicts the number of objects in low-earth orbit. Each curve represents a different combination of strategies (remove 0, 1 or 2 objects) taken by the two players (the US and the EU).	179

B.3	Evolution of the overall collision risk to important assets of the US (left) and the EU (right), for different actions taken by both players. We can see an exponential growth of the risks for cases where the players do not remove any objects.	180
B.4	Free-riding effect in the overall risk to important assets for non-active players China and Russia (both removing 0), for different combinations of actions taken by the US and the EU. Non-active players benefit from other players removing objects.	181
B.5	Equilibrium strategies for the sub-game {remove 0, remove 1} (left panel) and {remove 1, remove 2} (right panel) for a range of removal costs C_R . The y -axis shows the probability of each player (the US and the EU) removing 0 (left panel) or 1 (right panel) object. The x -axis shows the ratio between the cost of removal C_R and the cost of losing an important asset C_L (assuming $C_L = 1$).	184
B.6	Evolutionary dynamics of the sub-game {remove 0, remove 1} for different values of C_R . Stable attractors are indicated with \bullet and unstable attractors with \circ . The dotted line indicates the trajectory on which the mixed equilibrium \circ moves as C_R changes.	187
B.7	Evolutionary dynamics of the subgame {Remove 1, Remove 2} for different values of C_R . Stable attractors are indicated with \bullet and unstable attractors with \circ . The dotted line indicates the trajectory on which the mixed equilibrium \circ moves as C_R changes.	188
B.8	Equilibrium strategies for players the US, the EU and China (CN) of the game {remove 0, remove 1} for a range of removal costs C_R . The y -axis shows the probability of each player (the US, the EU, and CN) removing 0 objects, which is equivalent to one minus the probability of removing 1. The x -axis shows the ratio between the cost of removal C_R and the cost of losing an important asset C_L (assuming $C_L = 1$).	190
C.1	Validation for different sequences of <i>thresholds for removal</i> changed at different time points. Comparing simulation with approximation from the surrogate model. Especially the <i>pessimistic</i> approximation closely follows the actual simulation, thus our surrogate model is validated.	194

C.2 Validation for sequence of *thresholds for removal* - no-removal and [1,000, 3,000, 5,000, 8,000] changed after 50 years. Comparing simulation with approximation. Approximations closely follow the actual simulations of respective colour, thus validating our surrogate model. 195

List of Tables

2.1	Prisoner's dilemma.	15
3.1	Thesis main chapters overview: Comparison of modelling choices and assumptions.	40
4.1	List of notations used in the stochastic game model of the space debris removal.	62
4.2	Optimal single-agent static strategies for different parameter λ , where the strategy is fixed for the entire time horizon. For increasing λ (i.e. object removal gets more costly) the optimal strategy is to remove fewer objects (i.e. greater threshold for removal).	64
4.3	Payoff matrix for parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$ (i.e. row player owns 60% of all assets). In normal font are the row player's payoffs, in italic are the column player's payoffs. In bold are pure Nash equilibria.	65
4.4	Optimal multi-agent static strategies, where the solution concept is Nash equilibria. We show player A's and B's rewards, welfare and fairness for parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$ (i.e. player A owns 60% of all assets). There are three pure Nash equilibria and several mixed ones (we show only one mixed NE in the last column).	65
4.5	Optimal single-agent dynamic strategies for different parameter λ . For increasing λ (i.e., object removal gets more costly) the optimal strategy is to remove fewer objects (i.e., greater thresholds for removal) and the welfare decreases.	66

4.6	Learned single-agent dynamic strategies for different λ . Differences to the optimal strategies from Table 4.5 are shown in bold and differences in rewards are stated in the last column. We can successfully validate the learning process due to high similarity (very low differences) to the optimal strategies.	67
4.7	Optimal multi-agent dynamic strategies against fixed opponent for parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. We show optimal altruistic (altr) and selfish (self) strategies. In the first column we show the opponent (player B) fixed strategy. We state the rewards, welfare, fairness and price of anarchy between different solutions. We can see that fixed strategies can lead to very sub-optimal solutions.	69
4.8	Learned multi-agent dynamic strategies using Q-learning against Q-learning opponent with parameter $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. We show four different outcomes of the same setting. We can attain highly effective solutions using Q-learning for both players.	70
4.9	Learned multi-agent dynamic strategies using Q-learning against Q-learning opponent with parameter $\lambda = 0.1$ and different levels of assets share ξ_i . Starting from highly disproportional players in the top row to equally sized players in the middle row. High disproportion in the players' size attains high welfare but trades off for fairness, where the small sized player removes barely anything.	71
4.10	Comparison of different scenarios in terms of welfare ω and fairness ϕ for $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. We show combinations of single-agent, multi-agent, static and dynamic approaches which were obtained either by learning or by exact computation. In case there were multiple solutions for given scenario we present maximal and minimal values.	72
4.11	Comparison of different scenarios in terms of price of anarchy PoA for $\lambda = 0.1$ and share parameter $\xi_A = 0.6$. The codes of the scenarios are stated in Table 4.10. In bold we show PoA_d (static vs. dynamic) and in <i>italic</i> we show PoA_m (single-agent vs. multi-agent). Note that for example $PoA = 1.107$ means 10.7% inefficiency.	73
4.12	Comparing static and dynamic single-agent scenarios in terms of welfare ω and price of anarchy PoA_d for varying parameter λ . Note that for increasing λ (i.e., cost of removal becomes more expensive) the welfare decreases and the difference between a static and a dynamic scenario increases.	74

4.13	Comparison of static and dynamic multi-agent scenarios in terms of welfare ω for different levels of λ and share parameter ξ . The static scenario is obtained by computing Nash equilibria and the dynamic scenario is learned using Q-learning. In case of multiple solutions we state maximal and minimal values (multiple NE). Note that with increasing parameter λ (object removal becomes more expensive) and increasing ξ (the players become more equally sized) the welfare decreases. One can see the improvement in welfare of dynamic strategies compared to the static ones.	75
4.14	Comparison of single-agent and multi-agent static scenarios in terms of price of anarchy PoA_m for varying levels of share parameter ξ and parameter λ . We can observe the increasing inefficiency of solutions for increasing ξ (the players become more equally sized).	76
4.15	Comparison of single-agent dynamic vs. multi-agent dynamic and multi-agent static vs. multi-agent dynamic scenarios in terms of price of anarchy (PoA_m and PoA_d) for varying levels of share parameter ξ and parameter λ . Note that in the comparison of the single-agent dynamic vs. multi-agent dynamic scenarios for increasing parameter ξ (more equally sized players) the inefficiency increases.	77
6.1	List of notations used in BayesRQ.	112
A.1	Parameters for 4 different spacecraft classes (ultra-small, small, medium and large) and 3 launch scenarios (conservative, moderate and aggressive). Parameters μ, σ are defining the Gaussian launch function and parameter α the yearly mass launched to orbit.	172
A.2	Conservative: Proposed OneWeb-like constellation.	175
A.3	Moderate: Possible SpaceX-like constellation.	175
B.1	Payoff functions for the different strategies: <i>remove 2</i> , <i>remove 1</i> and <i>remove 0</i> . Cost of losing an asset C_L and cost of object removal C_R are parameters of the game, \hat{r} is the risk of collision obtained from the space debris simulator and T is time horizon.	178
B.2	Risk matrix for both players for each combination of strategies. The risks are the average cumulative risk of losing an asset over the course of 150 years. We show 95 % confidence intervals in the lower table.	182

- B.3 Payoff matrix for both players for $C_R = 0.003$. Best responses are in bold text, thus there are two pure Nash equilibria: {US 0, EU 1} and {US 1, EU 0}.183
- B.4 Risk matrix (top) and corresponding 95% confidence intervals (bottom) for a three-player (the US, *the EU* and **China**) two-action game, values shown in the font styles belonging to each player (normal, *italic* and **bold**, respectively).189

References

- Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–8.
- Albrecht, S. V. and Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95.
- Aleksandrov, M., Aziz, H., Gaspers, S., and Walsh, T. (2015). Online fair division: Analysing a food bank problem. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2540–2546.
- An, B., Kiekintveld, C., Shieh, E., Singh, S., Tambe, M., and Vorobeychik, Y. (2012). Security games with limited surveillance. *AAAI Conference on Artificial Intelligence*, pages 1241–1248.
- Anselmo, L., Rossi, A., Pardini, C., Cordelli, A., and Jehn, R. (2001). Effect of mitigation measures on the long-term evolution of the debris population. *Advances in Space Research*, 28(9):1427–1436.
- Arai, S., Sycara, K., and Payne, T. R. (2000). Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain. In *Pacific Rim International Conference on Artificial Intelligence*, pages 125–135.
- Arulkumaran, K., Deisenroth, M., Brundage, M., and Anthony Bharath, A. (2017). A brief survey of deep reinforcement learning. *IEEE Signal Processing Magazine*, 34(6):26–38.
- Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Annual Symposium on Foundations of Computer Science*, pages 322–331.

- Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, pages 1726–1734.
- Bamón, R. and Frayssé, J. (1985). Existence of Cournot equilibrium in large markets. *Econometrica*, 53(3):587–597.
- Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, MA.
- Bloembergen, D., Tuyls, K., Hennes, D., and Kaisers, M. (2015). Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, 53:659–697.
- Bosansky, B., Lisy, V., Lanctot, M., Cermak, J., and Winands, M. H. M. (2016). Algorithms for computing strategies in two-player simultaneous move games. *Artificial Intelligence*, 237:1–40.
- Bousquet, F., Barreteau, O., Le Page, C., Mullon, C., and Jacques, W. (1999). An environmental modelling approach: The use of multi-agent simulations. In *Advances in Environmental and Ecological Modelling*, pages 113–122. Elsevier, Paris, France.
- Bowling, M. and Veloso, M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Technical Report CMU-CS-00-165, DTIC Document.
- Bowling, M. and Veloso, M. (2001). Rational and convergent learning in stochastic games. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1021–1026.
- Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.
- Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews*, 38(2):156–172.
- Carrico, T., Carrico, J., Policastri, L., and Loucks, M. (2008). Investigating orbital debris events using numerical methods with full force model orbit propagation. *Advances in the Astronautical Sciences*, 130:407–426.
- Ciosek, K. A. and Whiteson, S. (2017). OFFER: Off-environment reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pages 1819–1825.

- Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 746–752.
- Conitzer, V. and Sandholm, T. (2003). AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *International Conference on Machine Learning (ICML)*, pages 83–90.
- Cristian, F., Dancey, B., and Dehn, J. (1996). Fault-tolerance in air traffic control systems. *ACM Transactions on Computer Systems (TOCS)*, 14(3):265–286.
- Darley, J. M. and Latané, B. (1968). Bystander intervention in emergencies: Diffusion of responsibility. *Journal of Personality and Social Psychology*, 8(4):377–383.
- Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. (2009). The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259.
- Davis, L. A. and Filip, L. (2015). How long does it take to develop and launch government satellite systems? Technical Report ATR-2015-00535, The Aerospace Corporation.
- Dearden, R., Friedman, N., and Russell, S. (1998). Bayesian Q-learning. *AAAI Conference on Artificial Intelligence*, pages 761–768.
- De Asis, K., Hernandez-Garcia, J., Holland, G., and Sutton, R. S. (2018). Multi-step reinforcement learning: A unifying algorithm. In *AAAI Conference on Artificial Intelligence*, pages 2902–2909.
- Diekmann, A. (1985). Volunteer’s dilemma. *Journal of Conflict Resolution*, 29(4):605–610.
- Dubey, P. (1986). Inefficiency of Nash equilibria. *Mathematics of Operations Research*, 11(1):1–8.
- Dubey, P., Haimanko, O., and Zapechelnyuk, A. (2006). Strategic complements and substitutes, and potential games. *Games and Economic Behavior*, 54(1):77 – 94.
- Dvoretzky, A. (1956). On stochastic approximation. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 39–55. University of California Press.

- Fang, F., Stone, P., and Tambe, M. (2015). When security games go green: designing defender strategies to prevent poaching and illegal fishing. *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2589–2595.
- Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018). Counterfactual multi-agent policy gradients. *AAAI Conference on Artificial Intelligence*, pages 2974–2982.
- Forshaw, J. L., Aglietti, G. S., Salmon, T., Retat, I., Roe, M., Burgess, C., Chabot, T., Pisseloup, A., Phipps, A., Bernal, C., et al. (2017). Final payload test results for the RemoveDebris active debris removal mission. *Acta Astronautica*, 138:326–342.
- Fudenberg, D. and Levine, D. K. (1998). *The Theory of Learning in Games*. The MIT Press, Cambridge, MA.
- Ganzfried, S. and Sandholm, T. (2011). Game theory-based opponent modeling in large imperfect-information games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2–6.
- Garcia, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480.
- Gaskett, C. (2003). Reinforcement learning under circumstances beyond its control. In *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*, pages 1–12.
- Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al. (2015). Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483.
- Greenwald, A. and Hall, K. (2003). Correlated Q-learning. In *International Conference on Machine Learning (ICML)*, pages 242–249.
- Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI Conference on Artificial Intelligence*, volume 4, pages 709–715.
- Hardin, G. (1968). The tragedy of the commons. *Science*, 162(3859):1243–1248.
- Harstad, B. (2012). Climate contracts: A game of emissions, investments, negotiations, and renegotiations. *The Review of Economic Studies*, 79(4):1527–1557.

- Heinrich, J., Lanctot, M., and Silver, D. (2015). Fictitious self-play in extensive-form games. In *International Conference on Machine Learning (ICML)*, volume 37, pages 805–813.
- Hennes, D., Claes, D., and Tuyls, K. (2013). Evolutionary advantage of reciprocity in collision avoidance. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Workshop on Autonomous Robots and Multirobot Systems (ARMS)*.
- Hennes, D., Jong, S. D., Tuyls, K., and Gal, Y. K. (2015). Metastrategies in large-scale bargaining settings. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(1):1–21.
- Hernandez-Leal, P., Kaisers, M., Baarslag, T., and Munoz de Cote, E. (2017). A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*.
- Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2018). Is multiagent deep reinforcement learning the answer or the question? A brief survey. *arXiv preprint arXiv:1810.05587*.
- Hong, Z.-W., Su, S.-Y., Shann, T.-Y., Chang, Y.-H., and Lee, C.-Y. (2018). A deep policy inference Q-network for multi-agent systems. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1388–1396.
- Hu, J. and Wellman, M. (2004). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4(6):1039–1069.
- Inter-Agency Space Debris Coordination Committee (2002). IADC Space debris mitigation guidelines. Technical report, Inter-Agency Space Debris Coordination Committee.
- Inter-Agency Space Debris Coordination Committee (2007). IADC Space debris mitigation guidelines. Technical Report Revision 1, IADC-02-01, Inter-Agency Space Debris Coordination Committee.
- International Organization for Standardization (2011). Space systems – Space debris mitigation requirements. Technical Report 24113, ISO.
- Izzo, D. (2012). PYGMO and PYKEP: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). Technical report, Advanced Concept Team - European Space Research and Technology Centre (ESTEC).

- Izzo, D., Getzner, I., Hennes, D., and Simões, L. F. (2015). Evolving solutions to TSP variants for active space debris removal. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1207–1214. ACM.
- Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). Convergence of stochastic iterative dynamic programming algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 703–710.
- Jain, M., Korzhyk, D., Vanek, O., Conitzer, V., Pechoucek, M., and Tambe, M. (2011). A double oracle algorithm for zero-sum security games on graphs. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 327–334.
- Johnson, N., L., Krisko, P., H., Liou, J.-C., and Anz-Meador, P., D. (2001). NASA’s new breakup model of EVOLVE 4.0. *Advances in Space Research*, 28(9):1377–1384.
- Jordan, P. R., Vorobeychik, Y., and Wellman, M. P. (2008). Searching for approximate equilibria in empirical games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1063–1070.
- Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.
- Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., and Levine, S. (2019). Model-based reinforcement learning for Atari. *arXiv preprint arXiv:1903.00374*.
- Katt, S., Oliehoek, F. A., and Amato, C. (2017). Learning in POMDPs with Monte Carlo tree search. In *International Conference on Machine Learning (ICML)*, volume 70, pages 1819–1827.
- Kessler, D. J. and Cour-Palais, B. G. (1978). Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research*, 83(A6):2637–2646.
- Kessler, D. J., Johnson, N. L., Liou, J.-C., and Matney, M. (2010). The Kessler syndrome: Implications to future space operations. *American Astronautical Society - Guidance and Control Conference*, pages 1–15.

- Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., and Tambe, M. (2009). Computing optimal randomized resource allocations for massive security games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 689–696.
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2018a). Learning robust policies when losing control. *Adaptive and Learning Agents workshop at AAMAS*.
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2018b). Towards learning to best respond when losing control. *European Workshop on Reinforcement Learning (EWRL)*, pages 1–11.
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2019). Robust temporal difference learning for critical domains. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 350–358.
- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Hennes, D., and Izzo, D. (2016a). Space debris removal: A game theoretic analysis. *Games*, 7(3):20.
- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Hennes, D., Izzo, D., Tuyls, K., and Summerer, L. (2016b). Game theoretic analysis of the space debris dilemma. Technical report, ESA Ariadna Study 15/8401.
- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Wittig, A., Sapera, A., and Izzo, D. (2018c). Space debris removal: Learning to cooperate and the price of anarchy. *Frontiers in Robotics and AI*, 5(54):22.
- Klima, R., Kiekintveld, C., and Lisy, V. (2014). Online learning methods for border patrol resource allocation. *Conference on Decision and Game Theory for Security (GAMESEC)*, pages 340–349.
- Klima, R., Lisy, V., and Kiekintveld, C. (2015). Combining online learning and equilibrium computation in security games. *Conference on Decision and Game Theory for Security (GAMESEC)*, pages 130–149.
- Klima, R., Tuyls, K., and Oliehoek, F. (2016c). Markov security games: Learning in spatial security problems. *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*, pages 1–8.

- Klima, R., Tuyls, K., and Oliehoek, F. (2018d). Model-based reinforcement learning under periodical observability. *AAAI Spring Symposium on Learning, Inference, and Control of Multi-Agent Systems*.
- Klinkrad, H. (2010). *Space debris*. Encyclopedia of Aerospace Engineering, Wiley Online Library.
- Klinkrad, H., Beltrami, P., Hauptmann, S., Martin, C., Sdunnus, H., Stokes, H., Walker, R., and Wilkinson, J. (2004). The ESA space debris mitigation handbook 2002. *Advances in Space Research*, 34(5):1251–1259.
- Klinkrad, H. and Johnson, N. (2009). Space debris environment remediation concepts. In *NASA DARPA International Conference on Orbital Debris Removal*, pages 8–10.
- Knight, J. C. (2002). Safety critical systems: challenges and directions. In *Proceedings of the 24th International Conference on Software Engineering*, pages 547–550. ACM.
- Knight, V., Komenda, I., and Griffiths, J. (2017). Measuring the price of anarchy in critical care unit interactions. *Journal of the Operational Research Society*, 68(6):630–642.
- Kober, J., Bagnell, A. J., and Peters, J. (2012). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274.
- Könönen, V. (2004). Asymmetric multiagent reinforcement learning. *Web Intelligence and Agent Systems: An international journal*, 2(2):105–121.
- Korzhyk, D., Yin, Z., Kiekintveld, C., Conitzer, V., and Tambe, M. (2011). Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41(2):297–327.
- Koutsoupias, E. and Papadimitriou, C. (1999). Worst-case equilibria. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science*, pages 404–413.
- Kukushkin, N. S. (1994). A fixed-point theorem for decreasing mappings. *Economics Letters*, 46(1):23–26.
- Kukushkin, N. S. (2004). Best response dynamics in finite games with additive aggregation. *Games and Economic Behavior*, 48(1):94–110.

- Kukushkin, N. S. (2005). Strategic supplements in games with polylinear interactions. *Russian Academy of Sciences*.
- Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40(E253):1–72.
- Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 464–473.
- Levhari, D. and Mirman, L. J. (1980). The great fish war: an example using a dynamic Cournot-Nash solution. *The Bell Journal of Economics*, 11(1):322–334.
- Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.
- Lewis, H., Swinerd, G., Newland, R., and Saunders, A. (2009). The fast debris evolution model. *Advances in Space Research*, 44(5):568–578.
- Lewis, H. G., White, A. E., Crowther, R., and Stokes, H. (2012). Synergy of debris mitigation and removal. *Acta Astronautica*, 81(1):62–68.
- Leyton-Brown, K. and Shoham, Y. (2008). Essentials of game theory: A concise multidisciplinary introduction. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2(1):1–88.
- Liou, J. C. (2011). An active debris removal parametric study for LEO environment remediation. *Advances in Space Research*, 47(11):1865–1876.
- Liou, J.-C., Anilkumar, A., Bastida, B., Hanada, T., Krag, H., Lewis, H., Raj, M., Rao, M., Rossi, A., and Sharma, R. (2013). Stability of the future LEO environment – an IADC comparison study. In *6th European Conference on Space Debris*, volume 723.
- Liou, J.-C. and Johnson, N. L. (2008). Instability of the present LEO satellite populations. *Advances in Space Research*, 41(7):1046–1053.
- Liou, J.-C. and Johnson, N. L. (2009). A sensitivity study of the effectiveness of active debris removal in LEO. *Acta Astronautica*, 64(2-3):236–243.

- Liou, J.-C., Johnson, N. L., and Hill, N. (2010). Controlling the growth of future LEO debris populations with active debris removal. *Acta Astronautica*, 66(5-6):648–653.
- Liou, J.-C., Kessler, D., Matney, M., and Stansbery, G. (2003). A new approach to evaluate collision probabilities among asteroids, comets, and kuiper belt objects. In *Lunar and Planetary Science Conference*, volume 34, page 1828.
- Littman, M. and Stone, P. (2001). Leading best-response strategies in repeated games. In *IJCAI Workshop on Economic Agents, Models, and Mechanisms*.
- Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 157–163.
- Littman, M. L. (2001). Friend-or-foe Q-learning in general-sum games. In *International Conference on Machine Learning (ICML)*, pages 322–328.
- Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning (ICML)*, pages 362–370.
- Liu, J., Xiao, Y., Li, S., Liang, W., and Chen, C. P. (2012). Cyber security and privacy issues in smart grids. *IEEE Communications Surveys & Tutorials*, 14(4):981–997.
- Lou, J., Smith, A. M., and Vorobeychik, Y. (2017). Multidefender security games. *IEEE Intelligent Systems*, 32(1):50–60.
- Maskin, E. and Tirole, J. (2001). Markov perfect equilibrium: I. observable actions. *Journal of Economic Theory*, 100(2):191–219.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *NIPS Deep Learning Workshop*.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529–533.
- Montesh, M. (2013). Rhino poaching: A new form of organised crime. Technical report, College of Law Research and Innovation Committee of the University of South Africa.

- Morimoto, J. and Doya, K. (2005). Robust reinforcement learning. *Neural computation*, 17(2):335–359.
- Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. (2016). Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1054–1062.
- NASA Orbital Debris Program Office (2007). Chinese anti-satellite test creates most severe orbital debris cloud in history. *Orbital Debris Quarterly News*, 11(2):2–3.
- NASA Orbital Debris Program Office (2009). Satellite collision leaves significant debris clouds. *Orbital Debris Quarterly News*, 13(2):1–2.
- NASA Orbital Debris Program Office (2011). International space station again dodges debris. *Orbital Debris Quarterly News*, 15(3):1–2.
- Nash, J. (1950). Equilibrium points in n -person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49.
- Nash, J. (1951). Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295.
- Nax, H. H. and Perc, M. (2015). Directional learning and the provisioning of public goods. *Scientific Reports*, 5:8010.
- Novshek, W. (1985). On the existence of Cournot equilibrium. *The Review of Economic Studies*, 52(1):85–98.
- Nowé, A., Vrancx, P., and De Hauwere, Y.-M. (2012). *Game theory and multi-agent reinforcement learning*, pages 441–470. Reinforcement Learning: State-of-the-Art, Springer Berlin Heidelberg.
- Omidshafiei, S., Papadimitriou, C., Piliouras, G., Tuyls, K., Rowland, M., Lespiau, J.-B., Czarnecki, W. M., Lanctot, M., Perolat, J., and Munos, R. (2019). α -rank: Multi-agent evaluation by evolution. *Scientific Reports*, 9(1):9937.
- Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434.
- Perc, M., Jordan, J. J., Rand, D. G., Wang, Z., Boccaletti, S., and Szolnoki, A. (2017). Statistical physics of human cooperation. *Physics Reports*, 687:1–51.

- Perolat, J., Leibo, J. Z., Zambaldi, V., Beattie, C., Tuyls, K., and Graepel, T. (2017). A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3646–3655.
- Phelps, S., Parsons, S., and McBurney, P. (2004). An evolutionary game-theoretic comparison of two double-auction market designs. In *International Workshop on Agent-Mediated Electronic Commerce*, volume 3435, pages 101–114.
- Pita, J., Jain, M., Tambe, M., Ordóñez, F., and Kraus, S. (2010). Robust solutions to Stackelberg games: Addressing bounded rationality and limited observations in human cognition. *Artificial Intelligence*, 174(15):1142–1171.
- Pita, J., Jain, M., Western, C., Portway, C., Tambe, M., Ordonez, F., Kraus, S., and Parachuri, P. (2008). Deployed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (Industry Track)*, pages 125–132.
- Polydoros, A. S. and Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173.
- Ponsen, M., Tuyls, K., Kaisers, M., and Ramon, J. (2009). An evolutionary game-theoretic analysis of poker strategies. *Entertainment Computing*, 1(1):39–45.
- Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., New York, NY.
- Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.
- Ross, S., Chaib-draa, B., and Pineau, J. (2007). Bayes-adaptive POMDPs. *Advances in Neural Information Processing Systems (NIPS)*, pages 1225–1232.
- Rossi, A., Alessi, E., Valsecchi, G., Lewis, H., Radtke, J., Bombardelli, C., and Bastida Virgili, B. (2017). A quantitative evaluation of the environmental impact of the mega constellations. In *European Conference on Space Debris*.
- Roughgarden, T. (2005). *Selfish routing and the price of anarchy*, volume 174. MIT press, Cambridge, MA.

- Roughgarden, T., Syrgkanis, V., and E., T. (2017). The price of anarchy in auctions. *Journal of Artificial Intelligence Research*, 1(59):59–101.
- Roughgarden, T. and Tardos, E. (2007). Introduction to the inefficiency of equilibria. *Algorithmic Game Theory*, 17:443–459.
- Ruan, S., Meirina, C., Yu, F., Pattipati, K. R., and Popp, R. L. (2005). Patrolling in a stochastic environment. Technical report, Electrical and Computer Engineering Department, University of Connecticut, Storrs.
- Rummery, G. A. and Niranjan, M. (1994). *Online Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering, Cambridge, UK.
- Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited.
- Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095–1100.
- Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. (2012). PROTECT: A deployed game theoretic system to protect the ports of the united states. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 13–20.
- Shoham, Y., Powers, R., and Grenager, T. (2003). Multi-agent reinforcement learning: A critical survey. Technical report, Stanford University.
- Shoham, Y., Powers, R., and Grenager, T. (2007). If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377.
- Shooman, M. L. (2003). *Reliability of computer systems and networks: fault tolerance, analysis, and design*. John Wiley & Sons, New York, NY.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

- Silver, D. and Veness, J. (2010). Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2164–2172.
- Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308.
- Singh, S. P., Barto, A. G., Grupen, R., and Connolly, C. (1994). Robust reinforcement learning in motion planning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 655–662.
- Smart, D. R. (1974). *Fixed point theorems*. Cambridge University Press, Cambridge, UK.
- Smith, J. M. and Price, G. R. (1973). The logic of animal conflict. *Nature*, 246(5427):15–18.
- Stackelberg, H. v. (2010). *Market structure and equilibrium*. Springer Science & Business Media.
- Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Grapel, T. (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2085–2087.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT press, Cambridge, MA.
- Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.
- Szepesvari, C. and Littman, M. L. (1997). Generalized Markov decision processes: Dynamic-programming and reinforcement-learning algorithms. Technical report, Brown University.
- Tahvonen, O. (1994). Carbon dioxide abatement as a differential game. *European Journal of Political Economy*, 10(1):685–705.
- Tambe, M. and An, B. (2012). Game theory for security: A real-world challenge problem for multiagent systems and beyond. In *AAAI Technical Report SS-12-03 Game Theory for Security, Sustainability and Health*.

- Thomas, C. D., Cameron, A., Green, R. E., Bakkenes, M., Beaumont, L. J., Collingham, Y. C., Erasmus, B. F., De Siqueira, M. F., Grainger, A., Hannah, L., et al. (2004). Extinction risk from climate change. *Nature*, 427(6970):145–148.
- Tsai, J., Rathi, S., Kiekintveld, C., Ordóñez, F., and Tambe, M. (2009). IRIS - A tool for strategic security allocation in transportation networks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (Industry Track)*, pages 37–44.
- Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine learning*, 16(3):185–202.
- Tuyls, K. and Nowé, A. (2005). Evolutionary game theory and multi-agent reinforcement learning. *The Knowledge Engineering Review*, 20(1):63–90.
- Tuyls, K., Perolat, J., Lanctot, M., Leibo, J. Z., and Graepel, T. (2018). A generalised method for empirical game theoretic analysis. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 77–85.
- Tuyls, K. and Stone, P. (2018). Multiagent learning paradigms. In *Multi-Agent Systems and Agreement Technologies*, volume 10767, pages 3–21. Springer.
- Tuyls, K. and Weiss, G. (2012). Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41–52.
- Vallado, D., Crawford, P., Hujsak, R., and Kelso, T. (2006). Revisiting spacetrack report# 3. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, volume 6753, pages 1–88.
- van Seijen, H., van Hasselt, H., Whiteson, S., and Wiering, M. (2009). A theoretical and empirical analysis of Expected Sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2009*, pages 177–184.
- von Neumann, J. and Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton University Press, Princeton, NJ.
- Walsh, W., Das, R., Tesauro, G., and Kephart, J. (2002). Analyzing complex strategic interactions in multi-agent systems. In *AAAI Conference on Artificial Intelligence - Workshop on Game-Theoretic and Decision-Theoretic Agents*.

- Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, UK.
- Weibull, J. W. (1997). *Evolutionary game theory*. MIT press, Cambridge, MA.
- Wellman, M. P. (2006). Methods for empirical game-theoretic analysis. In *AAAI Conference on Artificial Intelligence*, volume 2, pages 1552–1555.
- Wellman, M. P., Jordan, P. R., Kiekintveld, C., Miller, J., and Reeves, D. M. (2006). Empirical game-theoretic analysis of the TAC market games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Workshop on Game-Theoretic and Decision-Theoretic Agents*.
- Wellman, M. P. and Prakash, A. (2014). Empirical game-theoretic analysis of an adaptive cyber-defense scenario (preliminary report). In *Conference on Decision and Game Theory for Security (GAMESEC)*, pages 43–58.
- Yan, Y., Qian, Y., Sharif, H., and Tipper, D. (2013). A survey on smart grid communication infrastructures: Motivations, requirements and challenges. *IEEE communications surveys & tutorials*, 15(1):5–20.
- Zhou, K. and Doyle, J. C. (1998). *Essentials of robust control*, volume 104. Prentice hall, Upper Saddle River, NJ.

Publications

- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Hennes, D., and Izzo, D. (2016a). Space debris removal: A game theoretic analysis. *Games*, 7(3):20.
- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Hennes, D., Izzo, D., Tuyls, K., and Summerer, L. (2016b). Game theoretic analysis of the space debris dilemma. Technical report, ESA Ariadna Study 15/8401.
- Klima, R., Tuyls, K., and Oliehoek, F. (2016c). Markov security games: Learning in spatial security problems. *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*, pages 1–8.
- Klima, R., Bloembergen, D., Savani, R., Tuyls, K., Wittig, A., Saper, A., and Izzo, D. (2018c). Space debris removal: Learning to cooperate and the price of anarchy. *Frontiers in Robotics and AI*, 5(54):22.
- Klima, R., Tuyls, K., and Oliehoek, F. (2018d). Model-based reinforcement learning under periodical observability. *AAAI Spring Symposium on Learning, Inference, and Control of Multi-Agent Systems*.
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2018a). Learning robust policies when losing control. *Adaptive and Learning Agents workshop at AAMAS*.
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2018b). Towards learning to best respond when losing control. *European Workshop on Reinforcement Learning (EWRL)*, pages 1–11.
- Klima, R., Bloembergen, D., Kaisers, M., and Tuyls, K. (2019). Robust temporal difference learning for critical domains. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 350–358.



Space Debris Simulator Model

A.1 Simulating Space Debris Environment

The simulator is built on top of the Python scientific library PyKEP [Izzo, 2012]. PyKEP provides basic tools for astrodynamics research, including utilities to interface with online databases such as the SATCAT¹ and TLE (two-line element set)² databases, which provide orbital information on all active (not decayed) objects in the low Earth orbit (LEO) regime we are studying, including the orbital elements that uniquely identify an object’s orbit, and which are used for orbit propagation. These databases provide the input to our simulator. PyKEP also provides an implementation of the SGP4 satellite orbit propagator (via libsgp4³), which we use extensively in this work. To simulate the future development of space debris in low Earth orbit (LEO) we developed several sub-modules, including a collision model, a break-up model and two launch models. The simulation is stepped at a fixed time step (e.g., 5 days). We use the SGP4 propagator in PyKEP to update the

¹<https://celestrak.com>

²<https://www.space-track.org/>

³<https://github.com/dnwrnr/sgp4>

position of all orbital elements in our catalogue. At the end of each time step, the following procedures are executed:

- **decay** of objects: either natural decay caused by gravity drag or intentional decay caused by active mitigation,
- **collisions** between objects, producing new debris,
- **launches** of new spacecraft.

A.2 Collision and Breakup Model

To evaluate the probability of collision between objects we follow the *Cube* approach [Liou et al., 2003]. The Cube approach samples uniformly in time rather than space and is thus compatible with any orbital evolution simulation as it does not impose assumptions on the orbital geometry. This is particularly important in LEO, where orbital progression is significant in the considered time frame. We use the SGP4 [Vallado et al., 2006] orbital propagator to calculate the evolution of the ephemeris (i.e., position and velocity) of an orbiting object given its TLE description. Ephemerides of all objects are calculated at regular time intervals. Space is then partitioned by a regular 3D-lattice (forming cubes) and for any pair i, j of objects that fall into the same volume, the collision probability p_{ij} is calculated as follows:

$$P_{i,j} = s_i s_j V_{rel} \sigma U ,$$

where s_i, s_j are the spatial densities of object i and j in the *cube*, $\sigma = \pi(r_i + r_j)^2$ is the cross-sectional collision area, V_{rel} is the collision (relative) velocity of the two objects, and U is the volume of the cube. For each pair, a pseudo-random number x is generated from a uniform distribution over the interval $[0, 1)$; if $P_{i,j} > x$, a collision event is triggered.

We use the NASA standard breakup model [Johnson et al., 2001] to generate the population of fragments resulting from a collision event. The NASA/JSC breakup model is a widely accepted stochastic model of the fragmentation process of in-orbit collisions and explosions based on multiple ground-tests and radar observations of past events. The model provides distributions for size, mass and ejection velocity of the fragment population parametrised by the total mass and collision velocity of the parent objects. The number of fragments larger than a characteristic length-scale follows a power-law, the area-to-mass ratio follows a multivariate normal distribution, and the ejection velocity is sampled from a

log-normal distribution. For details we refer to the original paper of Johnson et al. [2001] as well as the description of the model in Klinkrad [2010]. For each sampled fragment, we create a new TLE entry using the fragment’s osculating elements, and add it to the population of objects being propagated. Although the breakup model also covers explosions as well as non-catastrophic collisions, we only consider catastrophic collisions (i.e., events leading to complete disintegration) in this work.

Decaying of Objects

Figure A.1 shows two examples (top and bottom row) of debris clouds resulting from collisions in our simulation. The debris cloud is plotted both directly after the collision (left) and after 10 years (right). Each dot represents one piece of debris; the size represents their mass. We can observe that a number of debris objects decay during these 10 years, in particular those with low altitudes. Atmospheric drag slows these object down, further reducing their altitude until they burn up in Earth’s atmosphere. An example of debris decay is given in Figure A.2. The object decays once it approaches the Earth’s surface, which is around 6371 km (Earth radius).

In the complex launch model we also consider an intentional decay of newly launched spacecrafts with mitigation guidelines implemented, we will discuss this in more detail in Section A.4.

A.3 Simple Launch Model

Firstly, we consider a simple future launch model, which we validate to be in line with the related work of Liou and Johnson Liou and Johnson [2009].

A.3.1 Repeating Launch Sequence

To simulate future launches of new satellites we assume a “business as usual” scenario based on past data. One can assume that future launches will differ from past launches by many factors, e.g., the mission purpose, the number of launches, their launch success rate and technology level, and the satellite’s ability to decay in given time frame, etc. However, as a first step of simplification we base our model on repeating a 10 year window from 2005 to 2015. From the SATCAT catalogue we filter all space objects introduced in this time window, excluding debris. For all these objects (both decayed and not decayed) we store the

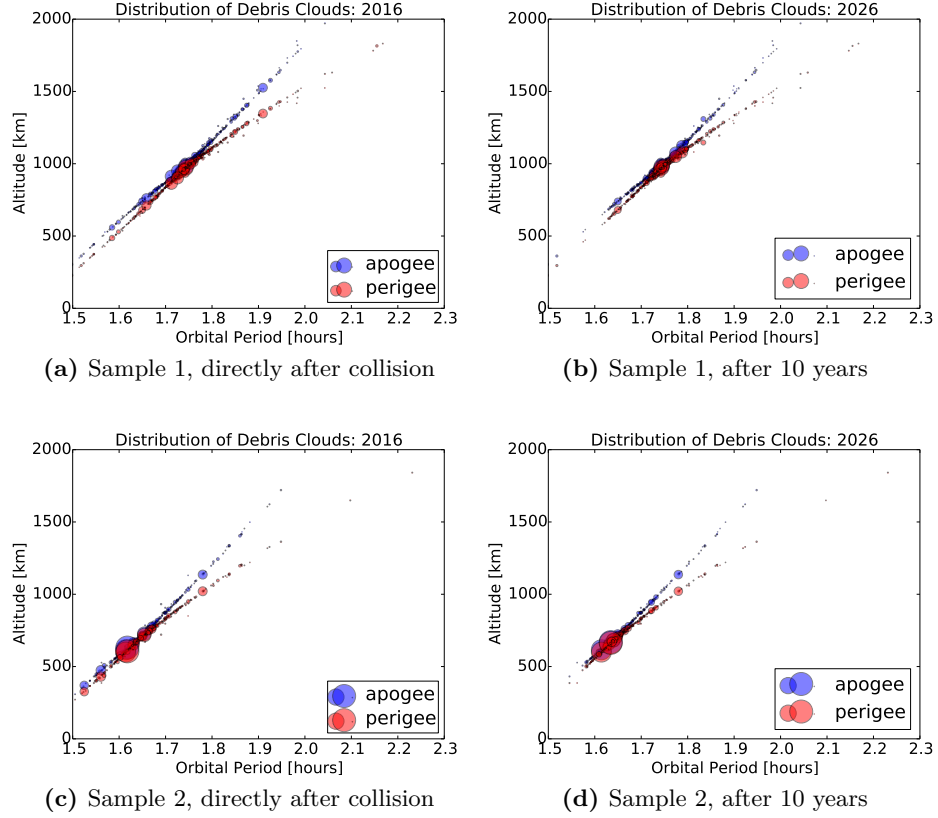


Figure A.1: Debris clouds from two sample collisions. Each dot is one piece of debris; the size represents their mass. Left figures show the clouds directly after the collision and right figures after 10 years. We can see how the objects in low altitudes decay.

TLE data (for the decayed objects we store the last TLE recorded). We then repeat this 10 year launch sequence and introduce each month all the objects that were launched exactly (a multiple of) ten years ago. We keep all the orbital elements the same, except for the inclination, which we sample randomly from the distribution of inclinations of all objects in the repeated sequence. This way, newly launched satellites will have slightly different orbits, as can be expected. Figure A.3 shows the distribution of orbital inclinations. We can see that the highest number of objects has an inclination of around 95° . We assume an increase over time in the number of launches due to technological development and changing needs. In addition to the 10 year repeating sequence, we increase the number of launches by 0.5% each year, by randomly sampling from the 10 year sequence. Note that

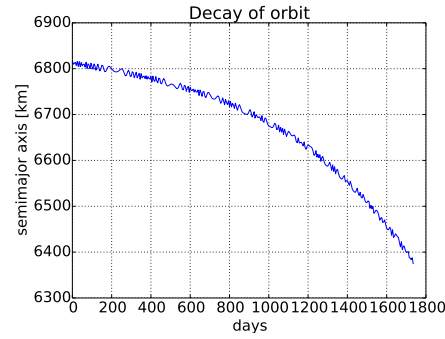


Figure A.2: Orbital decay of a sample debris object in Earth's atmosphere around 6371 km (Earth radius) caused by atmospheric drag.

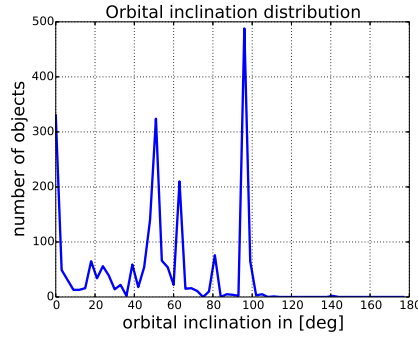


Figure A.3: Orbital inclination distribution of objects being launched in last 10 years.

each launch has a small probability of failing, due to the instability of some orbits resulting from the randomly sampled orbital inclination. Thus, some objects decay very soon after being launched, which can be thought of as for example unsuccessful launches, break-up during first stage, etc.

Figure A.4 shows how the orbital elements of all objects from 2005 to 2015 in LEO are distributed. We compare the orbital elements of debris (including rocket bodies) and important assets (active satellites). One can observe that a large number of debris and active satellites share the same inclination (close to 100) and eccentricity (close to 0). Note that Figure A.3 differs from Figure A.4a because it also includes already decayed objects.

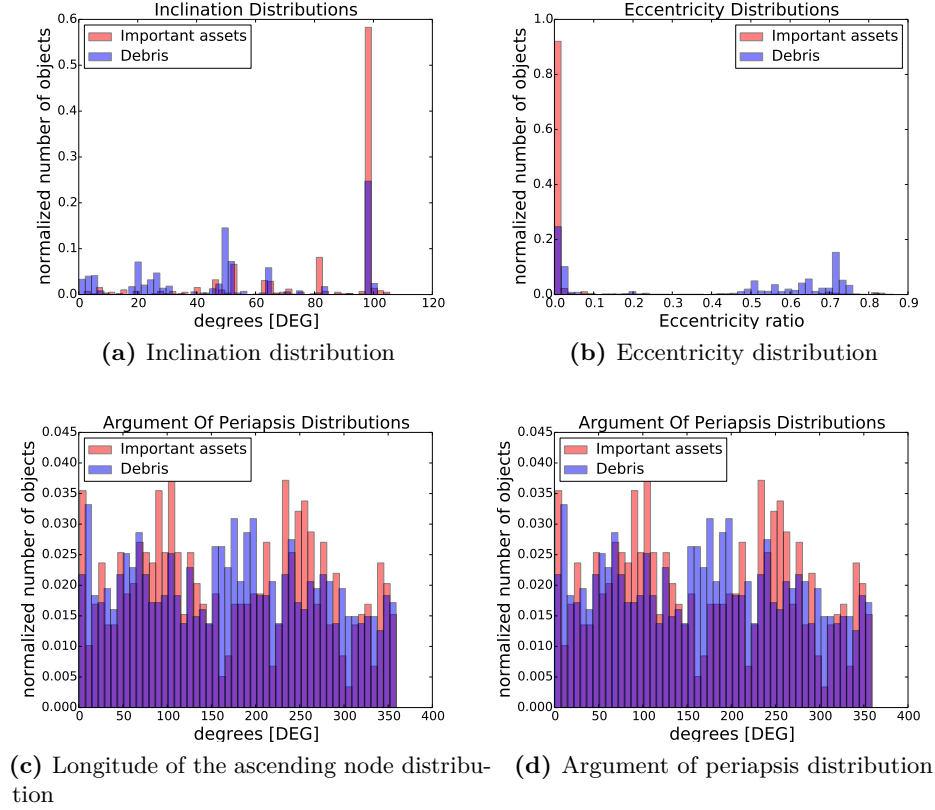


Figure A.4: Distribution of osculating elements (normalised to 1) of all important assets (red) and debris (blue) in LEO which are currently less than 10 years old. Important assets are active satellites. We can observe the most common inclination and eccentricity orbital parameters.

A.3.2 Validation

In order to validate our model we simulate the evolution of the total number of debris and compute the resulting spatial density in different altitude ranges for the next 150 years, and compare our findings to previously reported predictions. In Figure A.5a we show our prediction of spatial density in LEO, assuming no mitigation strategies. The three curves in Figure A.5a represent the situation in year 2015, and predictions for the years 2115 and 2165. One can observe that the highest spatial density is in the region around 800 kilometre altitude, caused by the Iridium-Kosmos collision (789 km) and the Chinese anti-satellite missile test causing the Fengyun-1C breakup (865 km). In our prediction, the spatial density

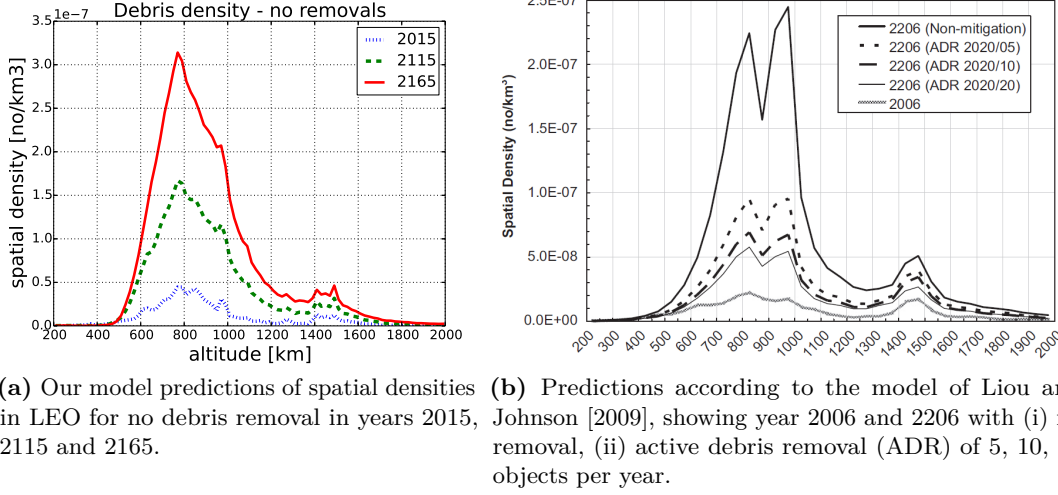


Figure A.5: Comparing spatial density predictions in LEO with related work of Liou and Johnson [2009]. Our model is in line with the related work.

increases significantly over time due to new collisions. We compare our findings with those reported previously by Liou and Johnson [2009], shown in Figure A.5b for different possible removal strategies. The non-mitigation scenario in Figure A.5b (for 2006 and 2106) shows a similar trend as the one observed in our model, with spatial density of debris increasing over time in particular in the altitude ranges around the Iridium-Kosmos collision and the Fengyun break-up. Small differences are likely due to different implementations, as the full details of the model of Liou and Johnson are not available.

A.4 Complex Launch Model

In this section we introduce a more complex type of launching, which moves from some of the simulator models described in the related work (e.g., Liou and Johnson [2009]), to a more realistic model. We now consider different future technological development speeds and implement recent mitigation rules [Klinkrad et al., 2004; Inter-Agency Space Debris Coordination Committee, 2002].

In order to simulate the future space environment, a crucial ingredient is the modelling of future launch activities into orbit. In Section A.3 we first considered a simple launching scheme, which employs a “business as usual” launch model that repeats the launch sequence

of a past period (e.g., one decade). The only adjustment made to accommodate technological advances is a potential speed-up of the launch sequence by scaling it to a shorter period in the future. The limitation with this modelling is that it does not account for disruptive innovation in space technology and space economy. The more complex launch model instead aims to provide finer control of the future scenarios. Clearly it is not possible to predict the future for the next century, and our launch model does not pretend to do that. Instead, our simulator is built to allow for a variety of possible future launch scenarios by adjusting various parameters. While this does not say anything about the probability of each scenario, it does allow to analyse their potential impact on the space environment if they were to happen.

As mentioned before the simulator is based on discrete time steps. To model future launches in this complex scenario, the key metric we use is the mass launched into LEO per year. This mass is continuously injected into the orbital environment by spreading it over four different classes of spacecraft, the relative distribution of which changes with time to model technological progress made.

A.4.1 Mass per Year

We choose to model the total mass launched per year into LEO using a quadratic function

$$M_{tot}(t) = M_{2000} \cdot (1 + \alpha(t - 2000)^2) \quad (\text{A.1})$$

with t in years AD and $M_{2000} = 200,000 \text{ kg}$ the total mass launched in the year 2000 used as a baseline. This function is purely heuristic and is meant to combine two effects: the increased launch capabilities becoming available, which increases the total mass launched per year, and miniaturisation of satellite technology, which reduces the need to launch large mass into LEO. As neither effect can be modelled with any certainty we opted for a simple function that has only one parameter α . The value of α allows to adjust the growth rate. Reasonable values would probably be around $\alpha = 10^{-4}$, leading to a doubling of the annual mass to orbit over the next 100 years, while $\alpha = 10^{-3}$ leads to a twenty-fold increase (see Figure A.6). Negative values correspond to a decrease in launched mass over time, which could happen either due to technological advances making large launch mass unnecessary or a marked downturn in space activity.

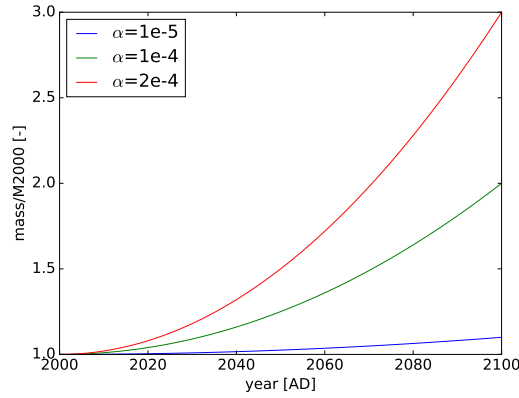


Figure A.6: *Mass to orbit* defining the total mass launched per year into low Earth orbit in relation to baseline mass launched in year 2000 (M_{2000}) for different values of α . For example the green curve represents a doubling of the annual mass to orbit over the next 100 years.

A.4.2 Spacecraft Classes

The following four classes of spacecraft are considered in our complex launch model:

- **Large satellites**, representative of the big communication and science satellites being actively launched and in common use today.
- **Medium satellites**, representative of newer science and technology demonstrators being developed and launched today and in the near future, e.g., for upcoming mega constellations.
- **Small satellites**, a group representative of cubesat type satellites which are being developed and tested today and may become increasingly attractive over the next decades.
- **Ultrasmall satellites**, a class of highly experimental nano-satellites, such as chipsats, envisioned to potentially become feasible in the future.

These classes are defined in terms of their attributes, including a typical mass and cost range for the satellite. More specifically, each class has the following attributes:

1. time dependent market share,

2. cost range,
3. mass range,
4. operational life time,
5. decay time.

With the exception of the market share function, all other attributes are represented as a range of values sampled uniformly each time a new spacecraft is launched. These attributes then remain assigned to the newly instantiated spacecraft for the remainder of its lifetime. They represent the total cost of the spacecraft including launch, the total mass of the spacecraft (ignoring any differences between dry and wet mass), the date of the end of operational life and the date when this spacecraft is scheduled to decay and burn up in the atmosphere following a controlled deorbiting manoeuvre.

The market share is the share of the total number of newly launched satellites at a given time that belongs to this particular class. It is the only attribute that is an explicit function of time. The idea behind it is that right now there are still many large traditional satellites being launched, but that number will decrease as cubesat technology will mature and smaller satellites can perform the same functions as their larger predecessors. To derive a market share function, we opted for a non-normalized Gaussian for each class:

$$g(t; \mu, \sigma) = \exp - \frac{(t - \mu)^2}{2\sigma^2}.$$

where t is measured in years since the year 2000. The two variables μ , indicating the center, and σ , indicating the width of the distribution, are chosen for each class and form a crucial part of the scenario definition. They are heuristically chosen such that the final market share function exhibits the desired trend for a given scenario. Intuitively the centre μ can be thought of as the point in time at which the production of that class of satellites peaks, while the width σ determines the slope and length of the build-up and decline of that class.

At each moment in time t , the probability p of a newly launched satellite belonging to class $x \in X$, i.e., the market share function for class x , is then given by the expression:

$$p_x(t) = \frac{g_x(t)}{\sum_{i \in X} g_i(t)} \tag{A.2}$$

where $g_i(t) = g(t; \mu_i, \sigma_i)$ is the Gaussian with the parameters for class i .

This definition of the market share function keeps the number of parameters defining the scenario sufficiently low, while providing enough flexibility to model different developments in the future. Examples for different scenarios and corresponding market share functions are given below.

A.4.3 Orbits

The remaining attributes that need to be decided when launching new spacecrafts are the actual orbits to inject the spacecrafts into. SGP4 uses averaged Keplerian orbital elements [Vallado et al., 2006] for its orbit representation. In that representation, and restricting ourselves to the LEO regime, we arrive at the following bounds for newly launched satellites:

- **Semi-major axis:** $a \in [300, 1200] + R_E$ km (distribution from current data)
- **Eccentricity:** $e \in [0.0, 0.5]$ (distribution from current data)
- **Inclination:** $i \in [0, 2\pi]$ (distribution from current data)
- **RAAN:** $\Omega \in [0, 2\pi]$ (uniform distribution)
- **Periapsis:** $\omega \in [0, 2\pi]$ (uniform distribution)
- **Mean anomaly:** $M \in [0, 2\pi]$ (uniform distribution)

The orbital elements semi-major axis (a), eccentricity (e), and inclination (i) are randomly chosen from the distribution of previously launched spacecraft. The rationale for just replicating the current distribution is that those orbital parameters represent orbits that are chosen for astrodynamical reasons such as sun-synchronous orbits or polar orbits. As these features are based on the underlying physics, they will not change in the future and the same orbits can reasonably be expected to remain relevant depending on the objective of the satellite. The right ascension of the ascending node (Ω), argument of periapsis (ω) and mean anomaly (M), instead, just represent orbital orientation and position of the spacecraft within the orbit, and are less relevant for the astrodynamic properties of the orbit. They are therefore chosen from a flat distribution. In our simulation, we obtain the distributions for semi-major axis, eccentricity, and inclination from the current space catalogue filtered for objects in the past 20 years and within the given bounds, as shown in Figure A.7.

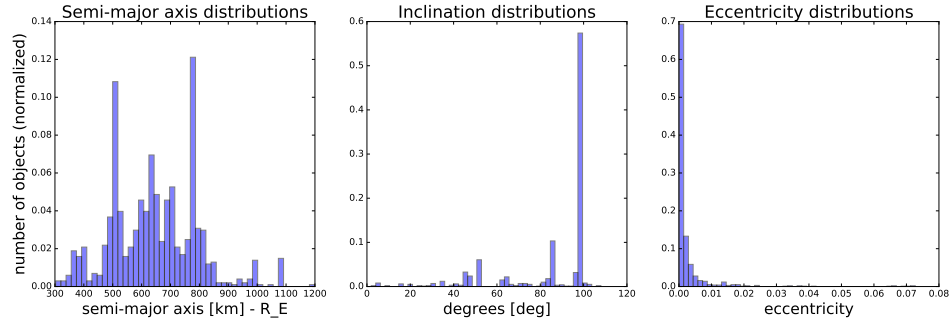


Figure A.7: Distributions of semi-major axis, inclination, and eccentricity of objects filtered from current space catalogue from last 20 years, from which we sample new orbital elements for objects to be launched. The remaining orbital elements are sampled uniformly at random from a given range.

The last input required for the SGP4 propagator is the so-called drag coefficient B^* nominally defined as

$$B^* = \frac{\rho_0 C_D A}{2m}$$

where $\rho_0 = 2.461 \cdot 10^{-5} \text{ kg/m}^3$ is the reference atmospheric density, C_D is the drag coefficient, and A/m is the area-to-mass ratio. As these values are typically not known exactly, in practice B^* is used to represent a range of non-conservative forces acting on the spacecraft. For real observations this parameter is typically fitted to provide the best agreement between SGP4 propagation and observation data. It is therefore possible to even find negative drag values in the satellite catalogue. As B^* is tightly related to the area to mass ratio A/m , and hence to spacecraft geometry, it is not possible to simply sample from previous distributions. As this parameter is used as a heuristic “catch-all” parameter in the SGP4 model, we simply set it to 0 for newly launched spacecraft. The justification for this is that for its active life a satellite will be maintained by its operator. This includes in particular orbit raising manoeuvres carried out regularly to maintain the operational orbit of a satellite. Similarly, as described above, our model assumes active end-of-life disposal of newly launched satellites by their respective operators. This eliminates the need for a drag term also during disposal. For debris fragments generated during in-orbit collisions, on the other hand, B^* does play a role in gradually decaying collision fragments. As the breakup model provides values for A/m , we simply assume a constant value of $C_D = 2.2$, typically

used for spacecraft where no other value has been determined experimentally, and compute B^* from that.

A.4.4 Launches

Instead of simulating individual launches, we regularly inject mass via averaged launches directly into the LEO environment. The justification for this is that while launches happen discretely, our simulation already ignores the rocket launcher itself, as well as initial commissioning and deployment phases of the satellites after being released by the launcher. Thus, trying to predict individual launches does not add anything to the accuracy of the simulation. A random set of new satellites is injected into orbit once per month, in our case the first time the simulation steps into a new month. The number of newly launched satellites is determined by the mass to orbit function evaluated at the current epoch. It provides the necessary information of how much mass to deliver to LEO each year, so dividing by 12 yields the newly launched mass to inject this month. The spacecraft class of newly launched spacecraft is sampled from the probability distribution given by the market share functions following Equation A.2. The properties of each spacecraft are selected randomly within the parameter ranges defined for each class of spacecraft. The orbits of the new spacecraft are chosen at random within the bounds specified by the global parameters of the simulation as detailed above. The spacecraft mass is then subtracted from the available launch mass. Note that even if the remaining launch mass is not sufficient, the spacecraft is still launched to avoid penalizing large spacecraft when dividing annual launch mass into monthly slices. As long as there is mass left, the process is repeated until all available launch mass has been used.

A.4.5 Future Launch Scenarios

To illustrate the complex launch model described above, we propose three scenarios that model the future nature of newly launched satellites. We chose these scenarios to illustrate three conceptually different developments in future launches with a time horizon of about 100 years. As mentioned before, we make no claim about how realistic these scenarios are, they are merely presented as possible developments in the future.

The three scenarios we propose are:

- **Conservative:** this scenario assumes little growth in space activity and is mostly “business as usual”. The total launched mass stays constant, and also technical progress

is slow. Relatively few and large spacecraft are being launched during most of the century.

- **Moderate:** this scenario assumes moderate growth in space activity. Total mass launched increases moderately, doubling over the next 100 years. Some technical progress is being made, but the market share of large and mid-size satellites remains significant also at the end of the century.
- **Aggressive:** this scenario assumes aggressive growth both in space activity and technological development. Note that the mass to orbit in this scenario is actually decreasing slightly by about 10% as spacecraft miniaturization technology is developing fast enough to keep up with increased demand. By the end of the century, the vast majority of newly launched spacecraft are cubesats and clouds of futuristic chipsats.

Table A.1 lists the parameters corresponding to these scenarios. Figure A.8 shows plots of the market share and total mass function for each scenario for illustration.

Decay of newly launched objects In contrast to the simple launch model (Section A.3), in this complex launch model the newly launched objects are assumed to follow the mitigation guidelines rules [Klinkrad et al., 2004]. Each newly launched object has a decay time assigned to it, this is the time after which the object is assumed to have decayed and it is removed from the simulation. This is to simulate modern satellites with active end-of-life mitigation techniques such as de-orbiting devices or graveyard orbit parking. While we do not simulate those de-orbiting actions explicitly, we do want to ensure that after the given lifetime objects do disappear from the catalogue.

A.4.6 Mega Constellations

In this complex launch model we have considered three classes of satellites for the future launch scenarios. An additional pseudo-class, *constellations*, could be used to represent medium size satellites in large constellations. These are launched at hand-picked dates defined in a scenario and into specific orbits. A constellation is built up over a certain time period $T_{deployment}$, during which some of the available launch mass each year is used for this task. Even after completion, the constellation continues to be maintained at a certain rate. This requires replacing new satellites into the constellation as old satellites are de-orbited.

	ultra-small	small	medium	large
cost range	[2k€, 1M€]	[1M€, 15M€]	[15M€, 40M€]	[40M€, 700M€]
mass range	[0.1 kg, 10 kg]	[10 kg, 100 kg]	[100 kg, 500 kg]	[500 kg, 5,000 kg]
operational time	[0.5 yr, 1 yr]	[0.5 yr, 2 yr]	[1 yr, 5 yr]	[10 yr, 20 yr]
decay time	[0.5 yr, 2 yr]	[1 yr, 7 yr]	[7 yr, 20 yr]	[10 yr, 25 yr]
Conservative				
μ	2200	2150	2060	2020
σ	50	50	40	60
α	0			
Moderate				
μ	2150	2090	2060	1970
σ	35	30	50	60
α	10^{-4}			
Aggressive				
μ	2150	2100	2040	1975
σ	50	40	30	60
α	-10^{-5}			

Table A.1: Parameters for 4 different spacecraft classes (ultra-small, small, medium and large) and 3 launch scenarios (conservative, moderate and aggressive). Parameters μ , σ are defining the Gaussian launch function and parameter α the yearly mass launched to orbit.

One could assume all constellations to be Walker Delta Pattern Constellations. That means a total number of satellites t is equally distributed over p circular ($e = 0$) orbital planes with fixed inclination i . Each plane's ascending node is rotated by $360^\circ/p$ relative to the previous plane (i.e., they are evenly spaced). The difference in the anomaly of a spacecraft in one plane and the next is given by $360^\circ f/t$. The notation used in the following is $i : t/p/f$. Constellation deployment can be simulated by each month launching $t/T_{\text{deployment}}$ satellites, where $T_{\text{deployment}}$ is the total deployment time in months. For simplicity, each satellite is injected in order, i.e., in the orbital plane following the last satellite with phase shift $360^\circ f/t$ in the anomaly. The launch mass is subtracted from the available launch mass that month before any other launches. Even if not enough launch mass is available, the constellation satellites are still injected. Constellation maintenance can then be simulated by “refreshing” constellation satellites that have reached their end-of-life with new ones. This is simply done by resetting their operational time and subtracting the corresponding launch mass from the available launch mass. This would ignore the

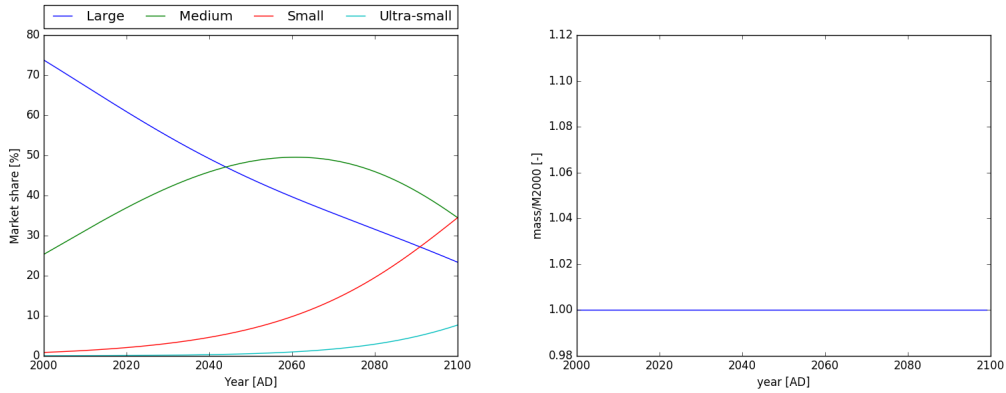
de-orbiting of the old satellite. Each constellation has the following attributes that can be defined in the scenario:

1. total number of satellites,
2. orbital distribution (some Walker configuration),
3. type of each satellite in constellation (with properties beyond orbit as for individual satellites),
4. construction time of the constellation,
5. life time of the constellation.

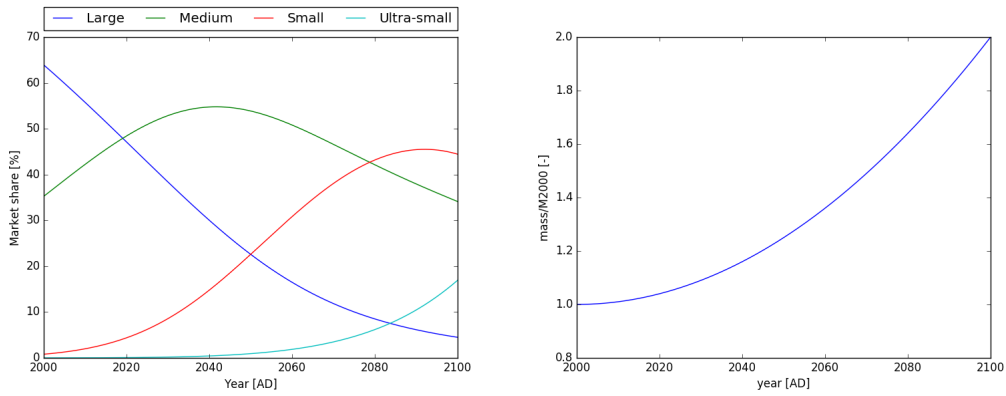
Various constellations are feasible. It is of course impossible to predict which will be launched when, but we can provide several options for various scenarios. The dates given are completely arbitrary and can be adjusted. For far future constellations beyond a reasonable planning horizon of about 20 years also the satellite size and weight should probably be adjusted. In Tables A.2 and A.3 we give two possible constellations, corresponding to a *conservative*⁴ and a *moderate*⁵ estimate for what is possible. In the context of the future launch scenarios, these should probably be deployed cumulatively, i.e., for a moderate scenario both the conservative and the moderate constellation should be deployed. Note that we do not consider this additional class of constellations in our experiments, but only state it here and leave it for future work.

⁴<http://www.bbc.com/news/science-environment-33268180>

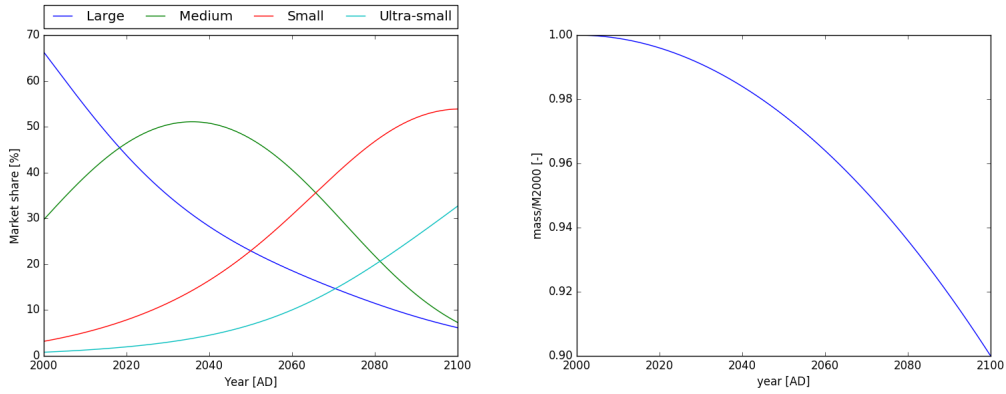
⁵<http://forum.nasaspaceflight.com/index.php?topic=41634.0>



(a) Conservative scenario



(b) Moderate scenario



(c) Aggressive scenario

Figure A.8: Market share and mass to orbit function for the conservative, moderate and aggressive launch scenarios as specified in Table A.1. The conservative scenario assumes “business as usual” with constant launch mass and slower technical progress, however the aggressive scenario assumes a fast technological development with emphasis on miniaturisation of spacecrafts.

Walker Constellation	90° : 720/18/1
Altitude	950 km
Inclination	90° (polar)
Number of satellites	720
Number of orbital planes	18
Satellite mass	175 kg
Satellite operational time	10 years
Satellite de-orbiting time	5 years
Satellite value	1M €
Construction starting date	January 2020
Construction completion date	January 2025
End of operation	January 2040

Table A.2: Conservative: Proposed OneWeb-like constellation.

Walker Constellation #1	53° : 1600/32/XX
Walker Constellation #2	53.8° : 1600/32/XX
Walker Constellation #3	74° : 400/8/XX
Altitude	1,150 km / 1,110 km / 1,130 km
Inclination	53° / 53.8° / 74°
Number of satellites	1600 / 1600 / 400
Number of orbital planes	32 / 32 / 8
Satellite mass	385 kg
Satellite operational time	7 years
Satellite de-orbiting time	1 years
Satellite value	2M €
Construction starting date	January 2020
Construction completion date	January 2025
End of operation	January 2035

Table A.3: Moderate: Possible SpaceX-like constellation.

B

The Space Debris Removal Problem as a Normal-form Game: Preliminary Work

In this appendix we introduce how the space debris removal dilemma can be modelled as a normal-form game in which the players are space actors, their actions are debris removal strategies, and the payoffs are derived from removal costs as well as collision risks, which are obtained by our space debris simulator. The strategic interaction results from the fact that debris removal by one agent may affect the collision risks to others as well. In this section we use the space debris simulator described in Appendix A with the simple launch model described in Appendix A.3.

B.1 Defining the Space Debris Removal Game

Players As a starting point we mainly focus our analysis on a two-player game, due to the amount of computation required to estimate the payoff function. Additionally, we discuss a game with three players. We consider players to be historically important actors; in the two player game choosing (1) the United States (US) represented by The National Aeronautics

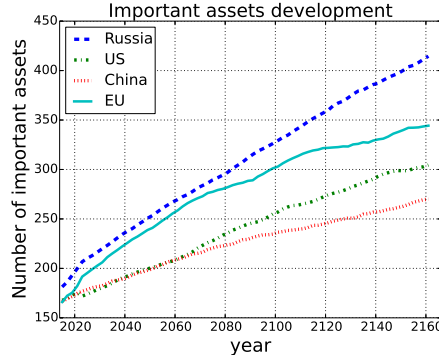


Figure B.1: Prediction of the development of the number of important assets (active satellites) for each player based on the assumption of a 0.5% yearly growth in future launches.

and Space Administration (NASA), and (2) the European Union (EU) represented by European Space Agency (ESA) together with all EU member states. Furthermore we assume a three player scenario where we add China (CN) as the third player; the fourth major space actor, Russia (Roscosmos), is not included in our game but does play a role in the simulator in terms of repeating past launch sequences.¹

Important assets For each player we store a list of *important assets*. Important assets are all active objects owned by that player which are not debris, and which have been launched in the last 10 years (we assume a 10 year life span of satellites). The list of important assets is continuously updated during the simulation. Figure B.1 shows an example of the development of important assets for each of the actors. One can observe that a small difference in the number of important assets at the beginning causes a big difference at the end of the projection due to the repetition of launches from the same sequence, combined with the 0.5% yearly increase.

Actions The players' actions are defined by the number of debris objects that will be removed each year. In our game, the players can remove 0, 1, or 2 risky objects every 2 years. We assume self-interested agents, meaning that each player first removes objects which directly threaten their important assets, and then removes objects which may potentially collide in general. The reasoning for the latter is that debris resulting from any collision

¹This choice is arbitrary, we expect similar results if Russia would be included as active player instead.

strategy	payoff function
remove 2	$-(\hat{r} \cdot C_L + T \cdot C_R)$
remove 1	$-(\hat{r} \cdot C_L + 0.5 \cdot T \cdot C_R)$
remove 0	$-(\hat{r} \cdot C_L)$

Table B.1: Payoff functions for the different strategies: *remove 2*, *remove 1* and *remove 0*. Cost of losing an asset C_L and cost of object removal C_R are parameters of the game, \hat{r} is the risk of collision obtained from the space debris simulator and T is time horizon.

may pose a potential future risk to player’s important assets. Therefore, removing any risky debris object (not only those that directly threaten important assets) may benefit all the players to some extent. Each player decides on his strategy at the beginning of the game, and does not change it later. Thus, we assume a one-shot normal form game.

Risks and payoffs During simulation we keep track of the risk of collision (defined as probability of collision by the space debris simulator, see Appendix A.2) to each player’s important assets. The cumulative sum of these risks is taken as the overall risk to each player under the simulated scenario. Subsequently, we derive payoffs from the costs of losing important assets, and the costs of object removal. These payoffs are computed by multiplying the player’s risk \hat{r} by the associated cost of losing an asset C_L , and adding the cost of removing one object each year C_R multiplied by the number of removed objects and the time horizon T . Specifically, Table B.1 lists the payoff functions that are used given the player’s strategy. Since the term $\hat{r} \cdot C_L$ is common to each strategy, we can assume without loss of generality that $C_L = 1$ (in arbitrary units) and focus only on C_R in the remainder.

B.2 Simulation Results and Projections

We use our simulator with the simple launch model to project the evolution of debris and collision risks with a time horizon of 150 years, i.e., the period 2016-2165, while repeating the launch history of 2006-2015 with a 0.5% yearly increase in number of launched satellites. We first focus on a 2-player 3-action game, with players the US and the EU, and the actions: *remove 0*, *1*, or *2* objects every two years as described above. For each combination of actions we average over 160 Monte-Carlo runs to account for randomness in the collision and break-up modules. Error margins are omitted in the figures for readability, but are

reported below in Table B.2.

B.2.1 Debris Evolution

Figure B.2 shows the evolution of objects in LEO for different combinations of strategies taken by the US and the EU. We observe an exponential growth trend without mitigation,

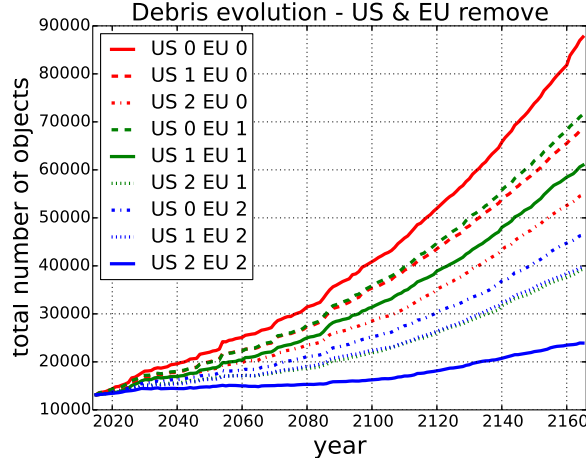


Figure B.2: Debris evolution for next 150 years considering different strategies. The y-axis depicts the number of objects in low-earth orbit. Each curve represents a different combination of strategies (remove 0, 1 or 2 objects) taken by the two players (the US and the EU).

in line with findings previously reported by Liou and Johnson [2009]. One can clearly see that removing risky objects has a positive effect as it leads to a much lower total number of objects in LEO. Note that when both players remove 2 objects every two years, this means that in total 300 objects are actively removed over the course of 150 years. In contrast, this leads to a reduction in total number of objects in LEO of over 60,000, due to a strong decrease in the number of collisions and resulting debris. Also note that the total number of active satellites (i.e., important assets, see Section B.1) in each scenario is less than 1,500, a small fraction of the total number of objects.

B.2.2 Risk Evolution

We now look at the potential risks to the players' important assets, that result from the debris evolution in LEO. Figure B.3a shows the evolution of the expected overall risk to

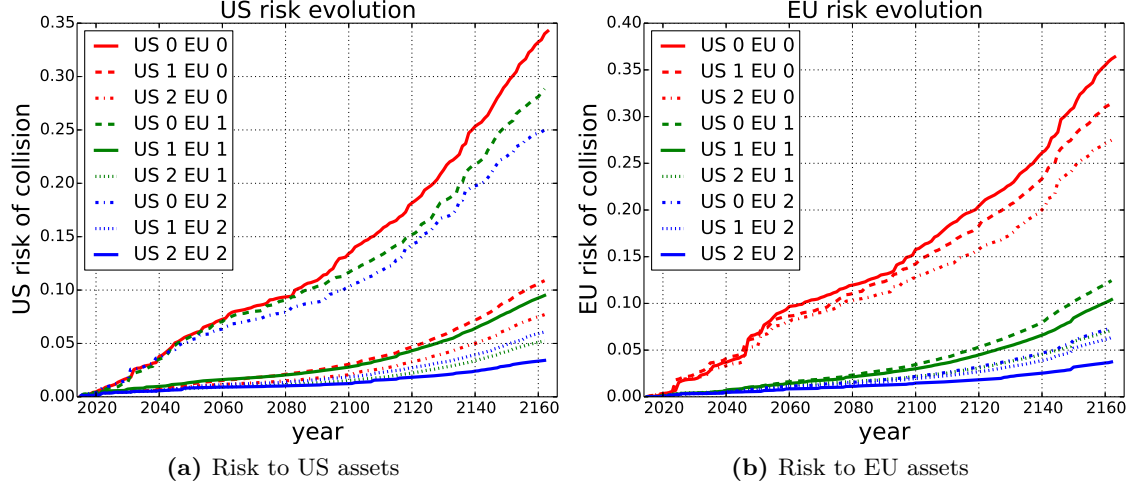


Figure B.3: Evolution of the overall collision risk to important assets of the US (left) and the EU (right), for different actions taken by both players. We can see an exponential growth of the risks for cases where the players do not remove any objects.

the US. One can observe that if the EU removes objects it helps the US as well. However, objects removed by the US have greater impact on their overall risk, which is explained by the fact that each player removes firstly the objects that threaten their important assets directly, and only then they remove objects that pose a risk in general. Therefore, we can see that the joint action $\{US\ 1, EU\ 0\}$ helps the US substantially more than $\{US\ 0, EU\ 2\}$, even though in the latter case more objects are removed in total. In Figure B.3b we can see the expected overall risk of losing important assets for the EU. We observe similar trends as in the previous figure: the EU is better off when they remove objects that directly threaten their assets. However, EU risks decrease, even when the EU does not remove any objects, but the US does remove. This means that, as expected, there is in fact a dilemma as each player benefits from mitigation efforts of others, without having to pay a cost (free-riding).

The free-riding effect can be observed as well when looking at the risk evolution for both China and Russia in Figures B.4a and B.4b. Even though these actors did not take part in mitigation in our scenario (essentially playing the fixed action of *remove 0*), they still benefit from a reduced risk to their important assets. One can notice an abrupt increase in the Chinese risks around the year 2080, which is eliminated when more objects are removed in total. The joint efforts of the US and the EU in fact remove the one object which causes

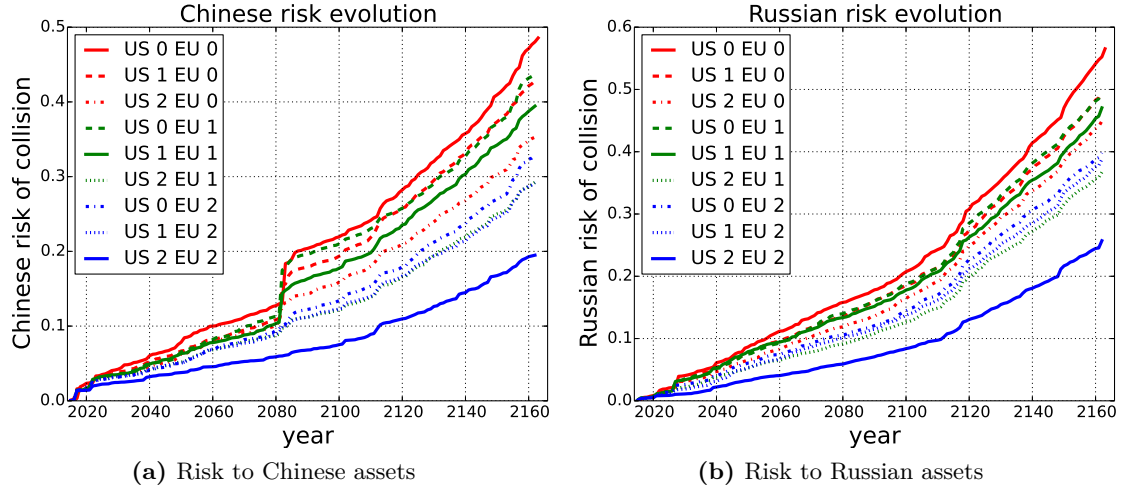


Figure B.4: Free-riding effect in the overall risk to important assets for non-active players China and Russia (both removing 0), for different combinations of actions taken by the US and the EU. Non-active players benefit from other players removing objects.

this high risk to the Chinese important assets. Note that this abrupt increase is persistent across simulation runs, caused by the deterministic nature of the orbital propagator.

B.3 Game Theoretic Analysis of Equilibria

We now introduce a game theoretic analysis of the space debris removal dilemma. First, we use the results reported in Section B.2 to derive a normal-form game representation of the two-player scenario. We then thoroughly analyse this game. Finally, we give an example of a three-player game. For an introduction on to game theoretic concepts see Section 2.1 where we also describe the Nash equilibrium which we use in this section.

B.3.1 Two Player Game

Using the simulation results of Section B.2, we can now construct a normal-form game representation of the two-player space debris removal dilemma. First, we construct a risk matrix, showing the overall risks to two players (the US and the EU) for each combination of actions. Then, we use this risk table together with the cost function defined in Table B.1 to derive payoff matrices for different removal costs C_R , and analyse all possible Nash

	<i>EU 2</i>	<i>EU 1</i>	<i>EU 0</i>
	risks		
US 2	0.03413, <i>0.03733</i>	0.05247, <i>0.07108</i>	0.07704, <i>0.27474</i>
US 1	0.06073, <i>0.06352</i>	0.09499, <i>0.10405</i>	0.10885, <i>0.31401</i>
US 0	0.25022, <i>0.07368</i>	0.28848, <i>0.12447</i>	0.34261, <i>0.36385</i>
	confidence intervals		
US 2	$\pm 0.00528, \pm 0.00563$	$\pm 0.00712, \pm 0.00785$	$\pm 0.00838, \pm 0.01874$
US 1	$\pm 0.00689, \pm 0.00767$	$\pm 0.00820, \pm 0.00938$	$\pm 0.00994, \pm 0.01954$
US 0	$\pm 0.01896, \pm 0.00786$	$\pm 0.01685, \pm 0.01061$	$\pm 0.01831, \pm 0.01859$

Table B.2: Risk matrix for both players for each combination of strategies. The risks are the average cumulative risk of losing an asset over the course of 150 years. We show 95 % confidence intervals in the lower table.

equilibria outcomes. This game has some interesting properties due to the payoff structure and asymmetry of the studied problem, which comes from different levels of space programs of the agents resulting in different number of assets in the orbits. There are other factors contributing to the asymmetric property of the game model such as position of some assets on orbits with higher density of space debris and therefore higher potential risks to these assets. Table B.2 shows the average cumulative risks accrued by both players, taken from the results in Figures B.3a and B.3b (time horizon 150 years, averaged over 160 MC runs for each scenario). A cumulative risk of 0.36385 for the EU in the no removal case can be interpreted as an expected loss of 0.36385 assets in total for the EU. The lower part of Table B.2 shows the 95% confidence intervals for these averages. Clearly, when no removal costs are taken into account, it is in the best interest of each player to remove as many debris as possible. However, one should assume non-zero removal costs. Using the cost functions of Table B.1 we can transform the risk matrix into a payoff matrix for any given cost C_R . Table B.3 shows an example payoff matrix for cost $C_R = 0.003$ (in arbitrary units, see Section B.1). The player's best responses are indicated in bold. One can see that there are two pure Nash equilibria in this scenario, $\{\text{US } 0, \text{EU } 1\}$ and $\{\text{US } 1, \text{EU } 0\}$. Moreover there is one mixed equilibrium in which the US and the EU mix between removing 1 and 0 with probability (0.488, 0.512) and (0.218, 0.782) respectively.

We can identify two interesting regions in the range of costs C_R . For very low costs,

	<i>EU 2</i>	<i>EU 1</i>	<i>EU 0</i>
US 2	-0.48413, -0.48733	-0.50247, -0.29608	-0.52704, - 0.27474
US 1	-0.28573, -0.51352	-0.31999, -0.32905	- 0.33385 , - 0.31401
US 0	- 0.25022 , -0.52368	- 0.28848 , - 0.34947	-0.34261, -0.36385

Table B.3: Payoff matrix for both players for $C_R = 0.003$. Best responses are in bold text, thus there are two pure Nash equilibria: {US 0, EU 1} and {US 1, EU 0}.

removing 0 will never be a best response for either player. Similarly, for high costs, removing 2 will never be a best response. Therefore we can focus on two sub-games defined by the removal action-pairs $\{0, 1\}$ and $\{1, 2\}$. We compute Nash equilibria for a range of C_R , and visualise the results in Figure B.5 for the sub-games $\{0, 1\}$ (left panel) and $\{1, 2\}$ (right panel). On the y -axis we have the probability of playing the first action in each sub-game (which equals 1 minus the probability of the second action) for the US (top) and the EU (bottom). The colours/line styles indicate the action pairs that make up the equilibria, e.g., the solid lines in Figure B.5 correspond to the pure Nash equilibria $(0, 0)$ (black) and $(1, 1)$ (red). The x -axis shows the ratio between the cost of removal C_R and the cost of losing an important asset C_L (assuming without loss of generality $C_L = 1$, as described in Section B.1). In both figures we see interesting transitions from the single Nash equilibrium at $(0, 0)$, to a situation where three equilibria exist at $(0, 1)$, $(1, 0)$ and one mixed, and finally back to a single pure equilibrium at $(1, 1)$. These transition phases also include a stage in which only one of the asymmetric pure equilibria at $(1, 0)$ or $(0, 1)$ exists. The existence of these asymmetric equilibria is interesting, and results from the asymmetry that is inherent in the risk matrix due to actors having different numbers of assets and in different orbits.

B.3.2 Strategic Substitutes and Existence of Pure Equilibrium

The games we construct here are finite strategic-form games. The celebrated result of Nash [1951] shows that every finite game possesses at least one Nash equilibrium in mixed strategies. While mixing makes a lot of sense in some settings, e.g., zero-sum games like poker and sports matches, in other settings pure strategy equilibria are more compelling. In this section we discuss properties, relevant for the games we construct, that guarantees the existence of pure equilibria.

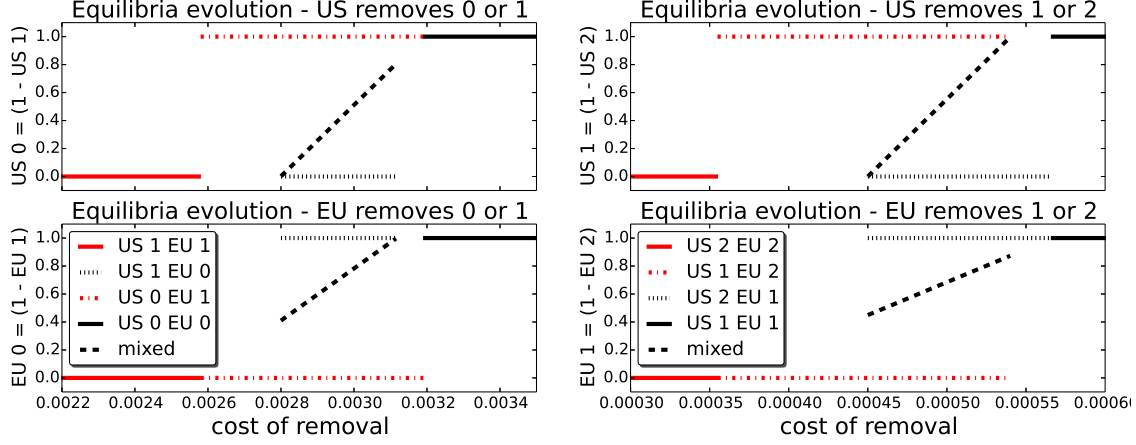


Figure B.5: Equilibrium strategies for the sub-game $\{\text{remove 0, remove 1}\}$ (left panel) and $\{\text{remove 1, remove 2}\}$ (right panel) for a range of removal costs C_R . The y -axis shows the probability of each player (the US and the EU) removing 0 (left panel) or 1 (right panel) object. The x -axis shows the ratio between the cost of removal C_R and the cost of losing an important asset C_L (assuming $C_L = 1$).

In general, active debris removal has a positive effect not only for the instigator of the removal but also for other players, and this is the cause of the dilemma that we are studying. In game-theoretic terminology, this suggests that we have games with a *weak strategic substitutes property*. The most well-known economic game with this property is Cournot oligopoly [Novshek, 1985; Bamón and Frayssé, 1985]. First we formally define the property. Our exposition is based on Dubey et al. [2006], but, for simplicity, is specialised to the setting of *finite* pure strategy sets. Denote the set of players by $N = \{1, 2, \dots, n\}$. Each player $i \in N$ has a finite pure strategy set Π_i that is a subset of non-negative real numbers, i.e., $\Pi_i \subset \mathbb{R}_{\geq 0}$. In our space debris removal games, Π_i can be thought of as the set of choices of how much debris player i removes, so in Table B.1, the three strategies *remove 0*, *remove 1*, *remove 2* would correspond to $\Pi_i = \{0, 1, 2\}$. Let Π denote the set of all pure strategy profiles, i.e., $\Pi := \Pi_1 \times \Pi_2 \times \dots \times \Pi_n$. The payoff (reward) function of player i is defined as $R_i : \Pi \rightarrow \mathbb{R}$.

For the purpose of stating known results on the existence of pure equilibria, we are going to assume that the payoff of player i depends only on his choice and the aggregate (i.e., sum) of the strategy choices of the other players. Formally, for any pure strategy

profile $\pi = (\pi_1, \pi_2, \dots, \pi_n) \in \Pi$, we denote by $\bar{\pi}_{-i}$ the *additive aggregate* of other players strategies, i.e.,

$$\bar{\pi}_{-i} = \sum_{j \in N \setminus \{i\}} \pi_j .$$

Then we write our restricted payoff function as $R_i(\pi_i, \bar{\pi}_{-i})$. For any choice $\pi_{-i} \in \prod_{j \in N \setminus \{i\}} \Pi_j$ as a product of finite pure strategy sets, the set $\beta_i(\bar{\pi}_{-i})$ of *best responses* of player i is given by

$$\beta_i(\bar{\pi}_{-i}) = \arg \max_{t \in \Pi_i} R_i(\pi_i, \bar{\pi}_{-i}) .$$

Recall that $\pi = (\pi_1, \pi_2, \dots, \pi_n) \in \Pi$ is a (pure) Nash equilibrium if

$$\pi_i \in \beta_i(\bar{\pi}_{-i})$$

for all $i \in N$. For a given player $i \in N$, we denote by $\bar{\Pi}_{-i}$ the set of all possible values of $\bar{\pi}_{-i}$, the additive aggregate of other players' strategies, i.e., $\bar{\Pi}_{-i} = \{\bar{\pi}_{-i} \mid \pi \in \Pi\}$. We say that a game like this has the *weak strategic substitutes property* if there exists a function $b_i : \bar{\Pi}_{-i}$ for these games with restricted payoffs functions such that:

- $b_i(x) \in \beta_i(x)$ for all $x \in \bar{\Pi}_{-i}$, [b_i selects a best response for i]
- $b_i(x) \leq b_i(y)$ whenever $x > y$. [b_i does not increase in $\bar{\pi}_{-i}$]

Such a game with the weak strategic substitutes property, and where payoffs depend only on one's own strategy and the sum of others' strategy, are known to always possess at least one pure strategy Nash equilibrium, which is shown via a potential-function type argument [Dubey et al., 2006; Kukushkin, 2004, 1994].

Notice that the weak strategic substitutes property can be defined as above even without the restriction that the payoffs are of the form $r_i(\pi_i, \bar{\pi}_{-i})$ for player i . However, in that case a pure equilibrium may not always exist. The games that we construct comprise payoffs that arise from (noisy) simulations and thus do not satisfy the restricted payoff form. However, the games we construct do either have the weak strategic substitutes property, or their violation of it is not statistically significant. Thus it is an interesting future direction to see if we can fit restricted payoff functions to closely approximate the empirical payoffs that arise from our simulations. We discuss this further below, where we also discuss slightly more general aggregation functions for defining restricted payoff functions that, along with the weak substitutes property, guarantee the existence of pure equilibria. First though

we note that when considering only two players, the restriction of the payoff functions is without loss of generality, and so we have the following.

Observation 1. *Any two-player game that has the weak strategic substitutes property admits a pure equilibrium.*

For example, in Table B.3, we can see that this game has the weak strategic substitutes property since, as the EU removes more (going from 0 to 1 to 2), the best responses of the US (as indicated by the boxes in Table B.3) is to weakly remove less (going from 1 to 0 to 0, respectively), and similarly for the best responses of the EU as the US changes pure strategy. This game has two pure equilibria (US 0, EU 1) and (US 1, EU 0) and one mixed equilibrium as one can also see in Figure B.5.

As mentioned above, for games with the weak strategic substitutes property, the existence of pure equilibria is known for a wider class of games than just those where the payoff of i depends on his strategy and the sum of the others'. This aggregation of players' strategies done by $\bar{\pi}_{-i}$ can in fact be an arbitrary linear combination rather than just a sum, and further can include linear combinations of *products* of strategies too; for full details see Kukushkin [2005]. Thus there is actually a lot of scope to fit payoff functions that meet these criteria for existence of a pure equilibrium due to strategic substitution and also are consistent with the payoff estimates that arise from our simulations. We leave this as an interesting direction for further work.

B.3.3 Evolutionary Dynamics

We now analyse the equilibria from evolutionary game theory perspective, where we look on the stability of different solutions. For introduction on evolutionary game theory and evolutionary stable strategies see Section 2.1.3.

Figure B.6 shows the directional field of the replicator dynamics for the sub-game {remove 0, remove 1} for different values of C_R corresponding to the different sets of equilibria observed in Figure B.5. The axes show the probability with which both players play the action "remove 0" (US 0 and EU 0). The arrows indicate the direction and magnitude of change. The replicator dynamics give insight into the stability of the different equilibria and their corresponding basins of attraction. We can conclude that the mixed Nash equilibria in panels (c) and (d) are unstable, as a small perturbation will cause the population to move towards one of the stable pure equilibria. Moreover we can see that the

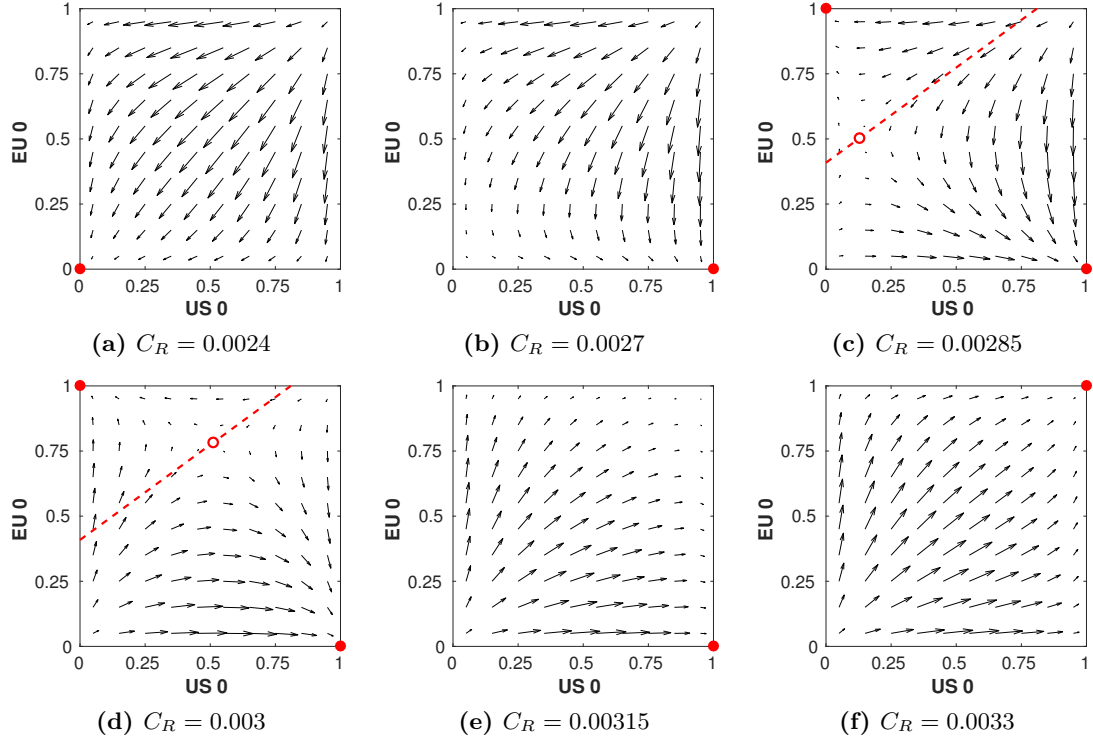


Figure B.6: Evolutionary dynamics of the sub-game $\{\text{remove } 0, \text{remove } 1\}$ for different values of C_R . Stable attractors are indicated with \bullet and unstable attractors with \circ . The dotted line indicates the trajectory on which the mixed equilibrium \circ moves as C_R changes.

basin of attraction for the pure equilibrium $\{\text{US } 0, \text{EU } 1\}$ (bottom right corner) is larger than for $\{\text{US } 1, \text{EU } 0\}$ indicating that this equilibrium is more likely to arise when both players iteratively optimise their strategy. This is of particular interest when full knowledge of the game is not available and the players need to *learn by interacting*, e.g., when space actors mutually adapt their policy based on an estimate of other actors' policies. In fact, the replicator dynamics are descriptive of various multi-agent learning processes, and as such studying these dynamics provides valuable insights in the context of *adaptive agents* as well [Bloembergen et al., 2015]. In Figure B.7 we show the directional field of the replicator dynamics for the sub-game $\{\text{remove } 1, \text{remove } 2\}$ for different values of C_R corresponding to the different sets of equilibria observed in right panel of Figure B.5.

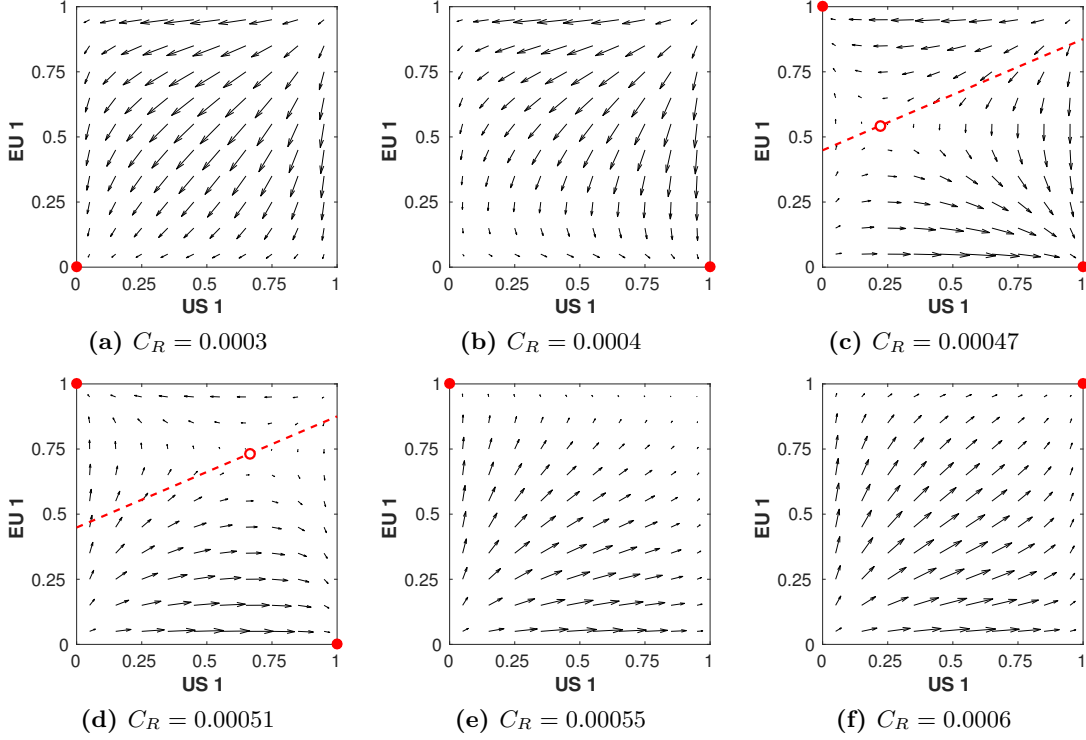


Figure B.7: Evolutionary dynamics of the subgame $\{\text{Remove 1, Remove 2}\}$ for different values of C_R . Stable attractors are indicated with \bullet and unstable attractors with \circ . The dotted line indicates the trajectory on which the mixed equilibrium \circ moves as C_R changes.

B.3.4 Three Player Game

So far we have only considered two active players. Here, we take a first step in analysing a larger game between three players (space actors): the US, the EU, and China (CN). We focus on the two-action sub-game $\{\text{remove 0, remove 1}\}$ only to facilitate analysis. Table B.4 shows the cumulative risks for all three players, averaged over 180 Monte Carlo runs, as well as the corresponding confidence intervals. The risks for each player are distinguished by different font styles. We can see that the risks for China are considerably higher than for the US or the EU, even though their total number of important assets is lower (see Figure B.1). This interesting result may be due to the specific orbits used by each of the players, some being more dense in terms of debris than others, which requires further investigation.

We can again convert the risk matrix into a payoff matrix using the payoff functions

		<i>EU 1</i>	<i>EU 0</i>
CN 1	US 1	0.07013, <i>0.08621</i> , 0.10162	0.09185, <i>0.31547</i> , 0.13320
	US 0	0.27373, <i>0.10294</i> , 0.12226	0.30759, <i>0.33980</i> , 0.15468
CN 0	US 1	0.09229, <i>0.10067</i> , 0.38774	0.11226, <i>0.32031</i> , 0.43121
	US 0	0.28510, <i>0.12539</i> , 0.43225	0.34400, <i>0.36335</i> , 0.49774
		<i>EU 1</i>	<i>EU 0</i>
CN 1	US 1	± 0.0061 , ± 0.0074 , $\pm \mathbf{0.0086}$	± 0.0071 , ± 0.0174 , $\pm \mathbf{0.0103}$
	US 0	± 0.0163 , ± 0.0080 , $\pm \mathbf{0.0098}$	± 0.0184 , ± 0.0189 , $\pm \mathbf{0.0110}$
CN 0	US 1	± 0.0077 , ± 0.0087 , $\pm \mathbf{0.0255}$	± 0.0093 , ± 0.0183 , $\pm \mathbf{0.0260}$
	US 0	± 0.0157 , ± 0.0099 , $\pm \mathbf{0.0275}$	± 0.0181 , ± 0.0180 , $\pm \mathbf{0.0279}$

Table B.4: Risk matrix (top) and corresponding 95% confidence intervals (bottom) for a three-player (the US, *the EU* and **China**) two-action game, values shown in the font styles belonging to each player (normal, *italic* and **bold**, respectively).

defined in Table B.1. In Figure B.8 we visualise the Nash equilibria for varying costs of removal C_R . At the left part of the figure the cost of removal is low, and therefore it is in the best interest of all three players to remove debris. However, for increasing costs the best response for the US becomes to stop removing, with a pure equilibrium (US 0, EU 1, CN 1). The reason that the US opts out first is due to their lower overall risk compared to the two other players. In contrast, the higher risks to China mean it is in their interest to keep removing, even when both the US and the EU have opted out. When the cost rises even further (the right side of the Figure B.8), we see that for none of the players removing is viable.

Although for most removal costs C_R the strategic substitute property discussed previously holds, there is a range of costs for which the property is violated. However, the payoff differences leading to this violation are not statistically significant and may be resolved by increasing the number of Monte Carlo samples of our simulation, which is left for future work.

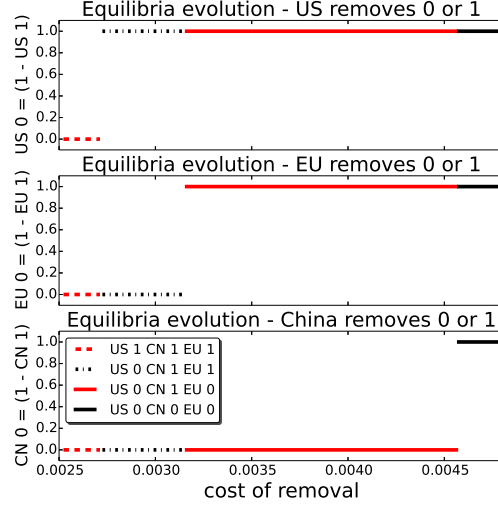


Figure B.8: Equilibrium strategies for players the US, the EU and China (CN) of the game {remove 0, remove 1} for a range of removal costs C_R . The y -axis shows the probability of each player (the US, the EU, and CN) removing 0 objects, which is equivalent to one minus the probability of removing 1. The x -axis shows the ratio between the cost of removal C_R and the cost of losing an important asset C_L (assuming $C_L = 1$).

B.4 Discussion

Using data from the space debris simulator with simple launch model we proposed a normal-form game, which we analysed by identifying Nash equilibria for different levels of cost of removals; although the costs of active debris removal are still prohibitively high at the moment they are expected to decrease with future technological development. Additionally, we investigated the strategic substitute property that appears in this type of game scenario, and which guarantees existence of a pure equilibrium under certain conditions. Although a mixed equilibrium exists for some costs as well, it is often more desirable to focus on pure equilibria. Specifically, in our scenario, it cannot be expected that space agencies will randomise over pure strategies to decide on their space debris removal policy. Another disadvantage of a mixed equilibrium in this game is its instability (as shown in Figure B.6), which is undesirable in the space debris scenario, where the choice of performed action has a huge impact on the earth orbit environment. For instance, we show how removing just one high risk debris object every two years can already substantially decrease the risk of

collision for active satellites. Additionally, removal of indirect collision risks is beneficial as well as it reduces the number of potential future on-orbit collisions.



Surrogate Model of Space Debris Evolution

C.1 Validation of Surrogate Model

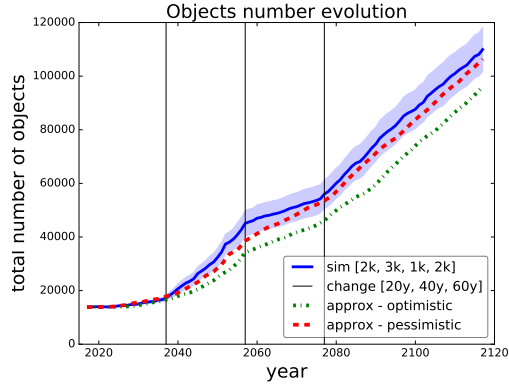
In order to validate our approximate surrogate model we compare outcomes of different settings of thresholds obtained from our surrogate model with the same settings obtained from the simulations of the full scale space debris model. In the following figures (Figures C.1a - C.1d) the blue solid curves represent the simulations and the red dashed and green dash-dot curves represent the surrogate model (pessimistic and optimistic shifting respectively). Similarly, in Figure C.2 the solid curves represent the simulations and the dashed curves represent the surrogate model. All the simulation curves have 95% confidence interval plotted with respective colour shading. We investigate several combinations of *threshold for removals* with the focus on switching between the thresholds over the time horizon. Comparing switching between different thresholds, which represent dynamic strategies, demonstrates the robustness of our surrogate model. The black horizontal lines show the points of changing the strategy (threshold). We only show several settings to validate our surrogate model because of the high computational demands, where every Monte Carlo run

takes on average 6 hours¹, i.e., each simulation curve taking approximately 600 hours if run on a single thread.

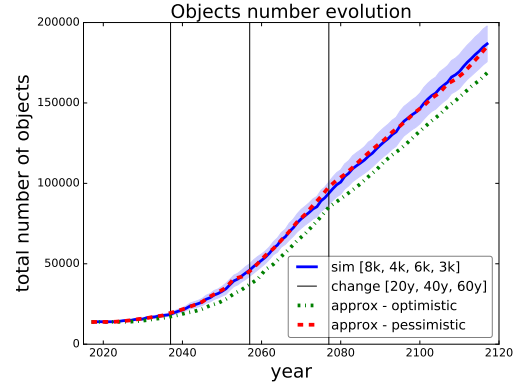
We start with the setting of thresholds $[2k, 3k, 1k, 2k]$ and switching points in years $[20y, 40y, 60y]$ in Figure C.1a. This setting represents a model where we use threshold 2,000 for the first 20 years, then we change to threshold 3,000 for 20 years, then to threshold 1,000 for 20 years and then threshold 2,000 for the rest of the time horizon, i.e., for another 40 years. Note the total time horizon is 100 years. We can see that the pessimistic approximation is most of the time within the simulation curve confidence bounds. In Figure C.1b one can see the simulation and approximation for setting $[8k, 4k, 6k, 3k]$ with the same intervals between switching. Again, the pessimistic approximation is within the confidence bounds of the simulation. In Figures C.1c and C.1d we set the switching times to 40 years and then 70 years. We can see that in the first figure the pessimistic approximation is most of the time within the confidence bounds. In the second figure we can see that at the end even the pessimistic approximation gets out of the bounds of the confidence interval, this is caused by an abrupt switch from no-removal strategy to threshold 1,000 strategy (two extreme strategies in our model). We will later restrict our model to non-abrupt switching between the threshold strategies due to this behaviour. Finally, in Figure C.2 we compare several settings of thresholds with only one switch after 50 years. All the curves start with no-removal strategy and after 50 years they switch to one of the thresholds 1,000, 3,000, 5,000, 8,000 and no-removal, i.e., continuing the no-removal strategy. One can observe that the abrupter the switch is the more the approximation curve deviates from the simulation curve. For example switching from no-removal to threshold 1,000 yields a significant deviation. On the other hand the approximation of switching from no-removal to threshold 8,000 stays within the confidence bounds of the respective simulation curve.

These experiments show that our approximate surrogate model – in the pessimistic setting – produces environment dynamics that fall within the 95% confidence interval of the real Monte Carlo simulations most of the times (except for abrupt switching). This successfully validates our methodology, approximating the actual simulations well. We can thus use the surrogate model to efficiently compute environment evolution and resulting costs for various debris removal strategies.

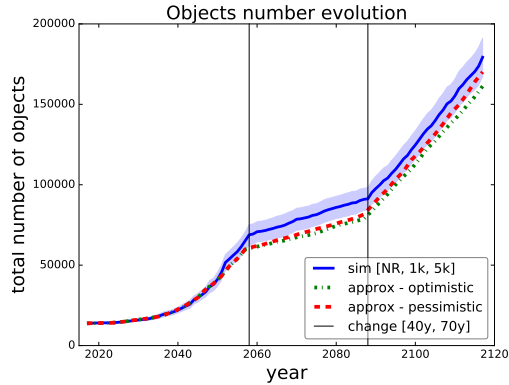
¹run on i7-2600 CPU @ 3.40GHz, 16GB of RAM.



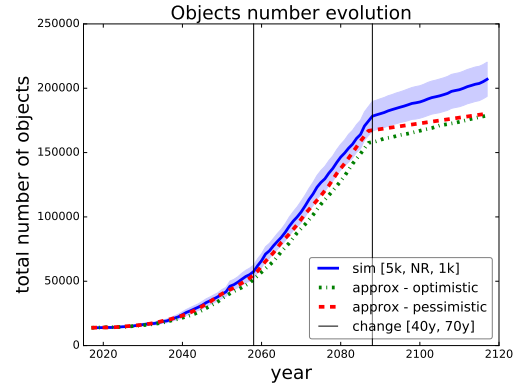
(a) thresholds 2,000, 3,000, 1,000 and 2,000 changed after 20, 40 and 60 years



(b) thresholds 8,000, 4,000, 6,000 and 3,000 changed after 20, 40 and 60 years



(c) thresholds no-removal, 1,000 and 5,000 changed after 40 and 70 years



(d) thresholds 5,000, no-removal and 1,000 changed after 40 and 70 years

Figure C.1: Validation for different sequences of *thresholds for removal* changed at different time points. Comparing simulation with approximation from the surrogate model. Especially the *pessimistic* approximation closely follows the actual simulation, thus our surrogate model is validated.

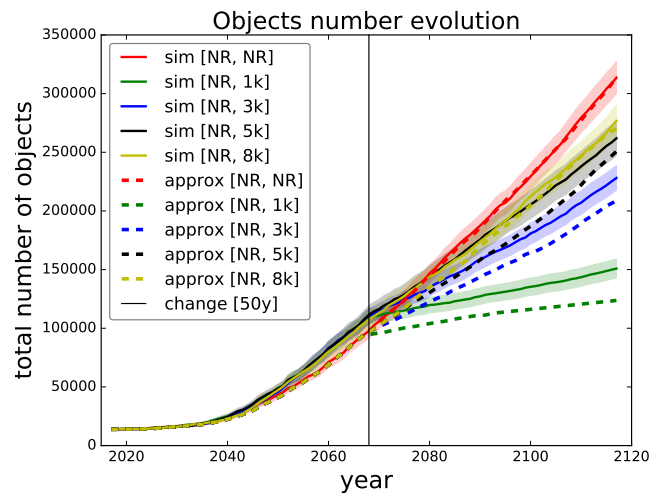


Figure C.2: Validation for sequence of *thresholds for removal* - no-removal and [1,000, 3,000, 5,000, 8,000] changed after 50 years. Comparing simulation with approximation. Approximations closely follow the actual simulations of respective colour, thus validating our surrogate model.